Virtuaalinen lomasaari

Vektorigrafiikan suunnittelu,
toteutus ja optimointi Flashissa

Viestintä
Graafinen suunnittelu
Opinnäytetyö
18.5.2010

Jenny Paatero

Metropolia

TIIVISTELMÄSIVU

Koulutusohjelma	Suuntautumisvaihtoehto
Viestintä	Graafinen suunnittelu

Tekijä
Jenny Paatero

Työn nimi
Virtuaalinen lomasaari - Vektorigrafiikan suunnittelu, toteutus ja optimointi Flashissa

Työn ohjaaja/ohjaajat
Arja Vuorio

Työn laji	Aika	Numeroidut sivut + liitteiden sivut
Opinnäytetyö	Toukokuu 2010	47+3

TIIVISTELMÄ

Opinnäytetyön tarkoituksena oli tutkia vektorigrafiikan piirtämistä ja optimointia Flashilla. Työn myötä syntyneet grafiikat tulevat Apaja Online Entertainment Oy:n uuden virtuaalitilan testikäyttöön.

Työn luova osa muodostui virtuaalisen lomasaaren konseptisuunnittelusta sekä uusien kuvaresurssien luomisesta noudattaen Aapeli.comin vakiintunutta visuaalista ilmettä. Opinnäytetyössä esiteltiin Flash-piirtämisen ja -animoinnin eri työvaiheet ja -välineet varhaisista luonnoksista lopullisiin kuviin.

Tutkimus käsitteli vertailevien kokeiden avulla erilaisten graafisten ratkaisujen vaikutusta Flash-esityksen toimivuuteen web-käytössä. Johtopäätöksiä käytetään lopullisen virtuaalitilan grafiikoiden käsittelyssä myöhemmin suoritettavaa testausta varten.

Opinnäytetyön tekniikoita ja tutkimustuloksia voidaan soveltaa yleisesti web-käyttöön tarkoitetun Flash-grafiikan toteutuksessa. Valmistunut visuaalinen esitys toimii esimerkkinä sekä tuotteistetusta kaksiulotteisesta grafiikasta että vektorigrafiikan erilaisista sovellustavoista.

Teos/Esitys/Produktio
Vektorigrafiikalla toteutettu virtuaalitila

Säilytyspaikka
Metropolia Ammattikorkeakoulun kirjasto, Tikkurilan toimipiste

Avainsanat
Flash, web-grafiikka, optimointi, virtuaalimaailma, animaatio, piirustustekniikka

Metropolia

Culture

Degree Programme in	Specialisation
Media	Graphic Design

Author
Jenny Paatero

Title
Virtual Holiday Island – Designing and optimizing vector graphics in Flash

Tutor(s)
Arja Vuorio

Type of Work	Date	Number of pages + appendices
Bachelor's Thesis	May 2010	47+3

This thesis studied the design and optimization of vector-based graphics in Flash. The resulting graphical resources will be used for testing Apaja Online Entertainment Oy's new virtual world project.

The creative part of this thesis consisted of concept design and the creation of new graphical resources according to Aapeli.com's established visual style. The thesis documents all stages and tools of drawing and animation in Flash from earliest concept sketches to final graphics.

The study used empirical tests to compare different graphical solutions and their effect on Flash presentations in web use. The results were used in the project itself to optimize graphics for future testing.

The techniques and research results of the thesis can be applied to all design of Flash-based graphics intended for web use. The final presentation is a showcase of productized 2D graphics and various ways of utilizing vector graphics for different ends.

Work / Performance / Project
Virtual space using vector graphics

Place of Storage
Metropolia Institute of Art and Design, Tikkurila

Keywords
Flash, web graphics, optimization, virtual world, animation, drawing technique

SISÄLLYS

1 JOHDANTO

Digitaalinen taide on ollut erityinen kiinnostuksen kohteeni jo yli kymmenen vuoden ajan. Vaikka olinkin jo kuusivuotiaana piirtänyt vanhalla Paintbrush-ohjelmalla alkeellisia kuvia, pääsin vasta vuosituhannen vaihteen tienoilla kokeilemaan ensimmäistä kertaa kunnollisia, nykyäänkin käytössä olevia kuvanmuokkausohjelmia. Opettelin itsenäisesti käyttämään sekä Photohop- että PaintShop Pro -ohjelmia, joilla piirtelin ja maalasin usein huvikseni. Digitaalisessa taiteessa minua kiehtoi etenkin sen teknisyys ja usein perinteisistä menetelmistä poikkeava ulkonäkö. Esimerkiksi Photoshopissa leikittelin usein erilaisilla suodatintoiminnoilla ja kehittelin omia automatisoituja efektisarjojani. Tein myös paljon pientä web-grafiikkaa, minkä myötä tutustuin digitaalisten kuvien ja etenkin web-käytön tuomiin teknisiin rajoituksiin.

Minua kiehtoi digitaalisissa kuvissa idea siitä, että niillä kaikilla on tiettyjä ominaisuuksia, jotka tekevät niistä ehdottoman hyviä tai huonoja. Nuo ominaisuudet ovat laskettavissa, eivätkä ole suoranaisesti sidoksissa kuvan taiteelliseen laatuun. Etenkin web-grafiikassa kuvan koko, muoto ja ominaisuudet ovat erottamattoman tärkeitä sen toimivuuden ja käyttötarkoituksen kannalta, joten esimerkiksi kahdesta samankaltaisesta digitaalisesta valokuvasta on konkreettisesti mitattavissa kumpi kuvista on johonkin tiettyyn tarkoitukseen parempi – esim. sivuston tausta- tai otsikkokuvaksi tarkoitetun tiedoston mitattavia arvoja ovat sen tiedostokoko tai resoluutio. Kummallakaan arvolla ei ole mitään tekemistä kuvan taiteellisen merkittävyyden kanssa, mutta ne vaikuttavat suoraan sen tekniseen käyttökelpoisuuteen.

Vasta Evtekissä tutustuin ensimmäistä kertaa vektorigrafiikkaan, mutta ihastuin nopeasti sen teknisyyteen. Pikseleihin perustuva digitaalimaalaaminen pyrkii nykyään imitoimaan luonnollisia välineitä niin paljon, että Photoshopillakin maalaillessa on helppo unohtaa, että käytössä on pelkkä maalia ja kangaspohjaa simuloiva ohjelma, kun taas Illustrator- tai Flash-ohjelmilla vektorikuvioiden teko on täysin erilaista, sillä se on alusta lähtien suorastaan matemaattista. Lisäksi

minua kiehtoi juuri se, että vektorikuvien tekninen laatu on aina täsmällisesti mitattavissa; esimerkiksi niiden monimutkaisuus ja efektien määrä voi vaikuttaa suoraan niiden käytettävyyteen web-grafiikkana.

Keväällä 2009 pääsin suorittamaan viiden kuukauden työharjoittelujaksoni Apaja Online Entertainment Oy:lla, joka tuottaa ja ylläpitää Aapeli-pelisivustoa (kansainväliset versiot kulkevat nimellä Playray). Pääsin ensimmäisestä päivästä lähtien oppimaan ja käyttämään Flashia kokeneiden ammattilaisten parissa. Tehtäviini kuului sekä Flashilla piirtäminen että piirrosten animointi, ja sain niiden kautta syvennettyä koulussa oppimaani tietoa käytännön kautta. Työharjoitteluni jälkeen jäin vielä kolmeksi kuukaudeksi Apajalle kesätöihin, ja kyselin jo silloin mahdollisuutta tehdä opinnäytetyöni heille hankkeistettuna. Tiedustelin asiasta ennen vuoden vaihdetta, ja sainkin kuulla lupaavasta projektista johon voisin osallistua.

Kyseessä olisi ns. virtuaalimaailma-alustan testaaminen, jossa pääsisin kokeilemaan erään kolmannen osapuolen kehittämää työkalua ja sen soveltamista Aapelin käyttöön. Käytännössä työni koostuisi testikäyttöön soveltuvan kaksiulotteisen virtuaalitilan suunnittelusta ja toteuttamisesta, eli pääsisin harjoittelemaan sekä konseptisuunnittelua että puhdasta piirtämistä. Alustan luonteesta ja Aapelin vakiintuneesta ilmeestä johtuen kaikki grafiikka tulisi tehdä Flashilla, mistä sainkin idean opinnäytetyöni tutkimuksellista osuutta varten.

Otin tavoitteekseni Flash-grafiikan tutkimuksen sekä piirustus- että optimointitekniikoiden kannalta. Aihe yhdistäisi sopivasti kaksi kiinnostuksen kohdettani; digitaalisen piirtämisen sekä web-käytön vaatiman viimeistelyn. Päätin tutkia etenkin Flashilla piirrettyä vektorigrafiikkaa pintaa syvemmältä tutustumalla sen rakenteeseen sekä erilaisiin välineisiin ja tekotapoihin. Arvioin, että aiheen tutkiminen antaisi minulle myös paremmat eväät työelämässä, sillä Flashin kaltaisen teknisen työvälineen ymmärtäminen mahdollistaisi varmasti tehtäviä sekä mainostoimistoissa että pelialan yrityksissä. Etenkin valmistunut testitila olisi varmasti komea visuaalinen esitys, jota voisin myös käyttää osana CV:täni. Projektin myötä saisin myös hiottua omaa työskentelyäni ammattimaisemmaksi ja tehokkaammaksi.

2 SUUNNITTELUN LÄHTÖKOHDAT

Virtuaalimaailmat ovat yksi nykyisiä verkkoyhteisöjen ja sosiaalisen median muotoja. Niiden tunnetuin ilmentymä on simuloitu teksti- tai kuvapohjainen ympäristö, jossa käyttäjä ohjaa yksilöllistä hahmoa, eli avataria, ja kykenee vaikuttamaan ympäristöön sekä kommunikoimaan toisten käyttäjien kanssa. Sosiaaliset virtuaalimaailmat ovat pääsääntöisesti pysyviä, eli ne ovat avoinna vuorokauden ajasta ja käyttäjien aikatauluista riippumatta.

Selaimissa toimivien verkkoyhteisöjen virtuaalitilojen yleisiä ilmenemismuotoja ovat erilaisten teemojen mukaiset graafiset keskusteluhuoneet (chat rooms), jotka simuloivat reaalielämän kohtaamispaikkoja, kuten puistoja, kahviloita, diskoja ja hotelleja. Joissakin palveluissa, esimerkiksi Habbo Hotellissa, käyttäjät voivat luoda omia tiloja ja sisustaa niitä haluamansa mukaisesti.

Virtuaalimaailmoihin kuuluu usein oma talousjärjestelmä. Käyttäjät voivat esimerkiksi ansaita tai ostaa palvelun virtuaalista valuuttaa, jolla he voivat vuorostaan hankkia virtuaalisia esineitä tai erityisominaisuuksia. Suosituimpien virtuaalimaailmojen talousjärjestelmät ulottuvat itse palvelun ulkopuolellekin: esimerkiksi Second Life -maailman käyttäjät voivat itse suunnitella ja toteuttaaa virtuaalisia esineitä tai palveluita, ja myydä niitä toisille Second Life -käyttäjille palvelun omaa tai reaalimaailman rahaa vastaan.

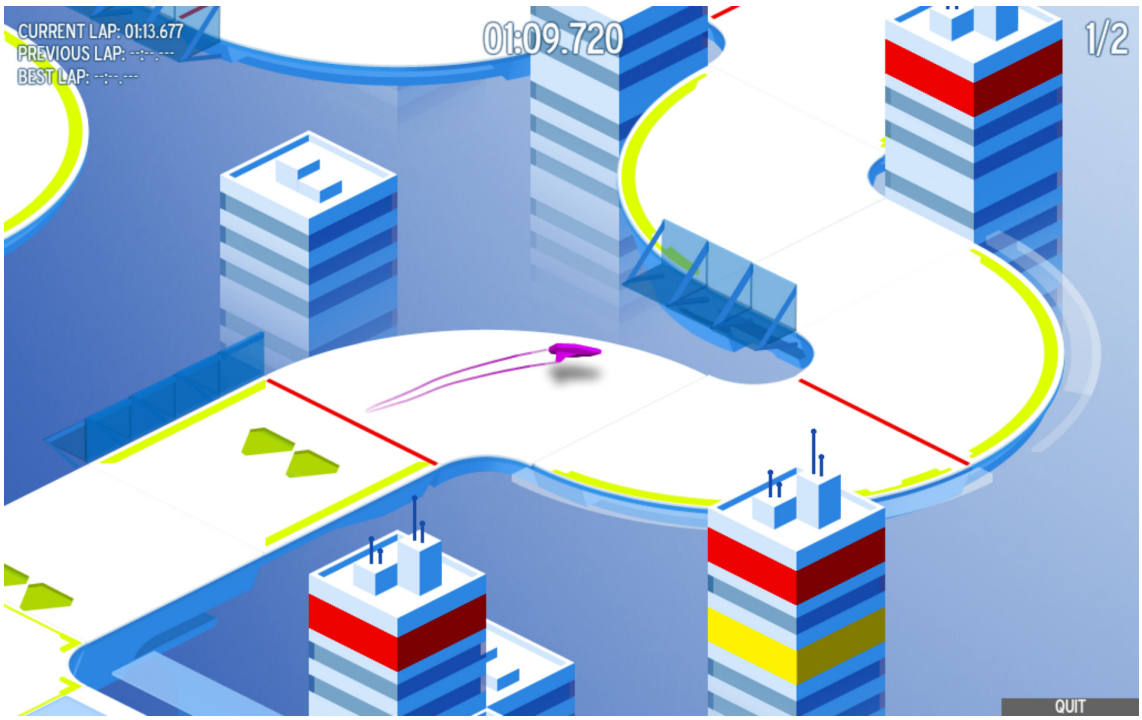
Virtuaalimaailmojen toiminnot jäljittelevät todellisen maailman lakeja, esimerkiksi paino- ja liikevoimaa ja ilmansuuntia. Kaksiulotteisissa maailmoissa luodaan illuusio kolmiulotteisuudesta ja korkeuseroista grafiikan avulla, esimerkiksi sijoittamalla tiettyjä elementtejä lomittain niin, että tilan läpi hahmo kulkeva näyttäisi olevan etualalla olevien asioiden peittämä, ja peittävän vuorostaan tilan taka-alalla olevat asiat.

2.1 Virtuaalihuoneprojektin esittely

Playforia oli Apajan vuonna 2006 Aapeli-pelisivustolleen suunnittelema virtuaalimaailmaprojekti, jonka kehitys lopetettiin pian teknisten ongelmien ja liiallisen monimutkaisuutensa takia. Projektin jäljiltä jäi kuitenkin mittava määrä virtuaalimaailmaa varten piirrettyä kaksiulotteista Flash-grafiikkaa, esimerkiksi täydellisiä huonekalusarjoja sekä sisä- ja ulkotilojen rakennuspalikoita.

Kuva 1. Viisikymmentäluku-henkinen Playforia-kahvila.

Monimonsu Oy on mediatoimisto Valvesta syntynyt pienyritys, joka on kehittänyt oman Flash-pohjaisen virtuaalimaailma-alustan. TrunkTech-nimisen alustan työkalut mahdollistavat ns. massivisten moninpelattavien sovellusten rakentamisen, ja sillä on tuotettu mm. Facebookissa toimiva kilpa-ajopeli Velocity sekä vielä työn alla oleva online-pelimaailma Eco-Rangers (Tweehouse Blog 2009). Alustan työkalujen avulla on mahdollista tuottaa monimutkaisia virtuaaliympäristöjä, joihin voidaan lisätä erilaisia sosiaalisia ja pelillisiä elementtejä. Lisäksi alusta on, tekijöidensä mukaan, yhdistettävissä olemassa olevien palveluiden tietokantoihin, joten se olisi mahdollista mm. liittää yhteen Aapelin hahmokaupan kanssa.



Kuva 2. Velocity on TrunkTech-työkalun avulla tehty Facebookissa toimiva kilpa-ajopeli.

Tehtävänäni on tutkia TrunkTech-työkalun soveltamista Apajan käyttöön, ja etenkin kartoittaa mahdollisuuksia hyödyntää siinä aikaisemmasta Playforia-projektista talteen jäänyttä kuvamateriaalia. Tutkimusta varten tarvitsisin kuitenkin uuden, erityisesti työkalua varten suunnitellun virtuaalitilan pohjan, jotta voisin sen avulla koota järkevän testiympäristön Playforia-esineille.

2.2 Haasteet

Virtuaalimaailmaprojektit eivät ole vailla haasteita, sillä niiden tuottaminen on vaativaa ja kilpailijoita on lukuisia. Tällä hetkellä kuluttajien käytös jälleen kulkemassa uuteen suuntaan, sillä vielä kolmisen vuotta sitten suosionsa huipulla olleet monimutkaiset 3D-maailmat ovat sosiaalisena mediana häviämässä kevyille selainpeleille. Esimerkiksi Facebookin Farmville ja muut yksinkertaiset, lyhyiden rupeamien pelit ja pelien kaltaiset sovellukset ovat suosittuja niin nuorten kuin aikuistenkin keskuudessa. Nopeasti saavutettavat ja helppokäyttöiset, sosiaalisiin sivustoihin liitettävät pelit ja virtuaalitilat yhdistävät ystävien kanssa keskustelun sekä kevyen satunnaispelaamisen, ja niissä on myös mahdollista vertailla pelituloksia tai antaa lahjoja ystävien kesken. Immersiivisen 3D-kokemuksen sijasta peruskäyttäjille riittääkin yksinkertaisempi virtuaaliympäristö, kunhan siinä on toimivat välineet kommunikoinnille ja hauskanpidolle (Terdiman 2010).

Kuva 3. Farmville on yksinkertainen, mutta lähes absurdin suosittu Facebook-peli.

Yksinkertaiset selainpelit ja -tilat ovat myös tekijöilleen hyödyllisiä, sillä sellaisen rakentaminen ja ylläpito on nopeampaa ja edullisempaa kuin täydellisen realistisen virtuaalimaailman tuottaminen. Monimutkaisten ominaisuuksien tai grafiikan sijasta tekijät voivat keskittyä sisällön tuottamiseen, sillä ilman jatkuvia päivityksiä käyttäjät menettävät nopeasti mielenkiintonsa palveluun.

Aapelin virtuaalimaailman uudelleenrakentamisessa ei olisi kyse kokonaan uuden palvelun luomisesta tyhjästä, vaan lisätoiminnan tuottamisesta jo olemassa olevalle verkkoyhteisölle. Tarvetta ei siis olisi kokonaisen uuden maailman rakentamiselle.

Aapelin hahmokauppa, josta käyttäjät voivat ostaa pelimerkkien ja -ominaisuuksien lisäksi vaatteita ja asusteita omien paperinukkehahmojensa koristeluun, on suosittu käyttäjien keskuudessa sekä Suomessa että ulkomailla. Hahmokauppaan lisätyt uudet tuotekokoelmat lisäävät kävijämääriä ja myyntiä koko sivustolla, minkä huomasin useaan otteeseen työharjoitteluni aikana. Se on merkki siitä, että käyttäjät ovat kiinnostuneita avatariensa ainutlaatuisuudesta ja valmiita maksamaan suuriakin summia pelkistä virtuaaliesineistä. Luonteva jatke avatar-hahmojen koristelulle olisi hahmon oma huone, johon olisi mahdollista ansaita tai ostaa Aapelin hahmokaupasta huonekaluja ja muita koristeita. Huoneeseen voisi esimerkiksi kutsua ystävien hahmoja kyläilemään, ja se toimisi eräänlaisena yksityisenä chat-huoneena. Huoneeseen voisi myös lisätä interaktiivisia elementtejä, esimerkiksi koneita, joista olisi mahdollista käynnistää Aapelin selainpelejä pelattavaksi huoneessa läsnä olevien vieraiden kanssa.

Kuva 4. Aapelin käyttäjät voivat hankkia itselleen hahmokaupasta vaatteita ja asusteita hahmoaan varten. Kaupasta voi lisäksi ostaa teemoja, taustakuvia sekä kelluvia esineitä koko sivun koristeluun.

Voidakseen kilpailla esimerkiksi Facebook-pelien kanssa uuden virtuaalitilan tulisi teknisiltä ominaisuuksiltaan olla mahdollisimman yksinkertainen ja helposti saavutettavissa. Tämä tarkoittaa sitä, että sen tulisi olla nopeasti latautuva ja vaatia mahdollisimman vähän resursseja käyttäjän laitteistolta, mahdollistaen jopa käytön nykyaikaisilla mobiililaitteilla. Käyttöliittymältään sen tulisi olla intuitiivinen, helposti omaksuttava ja muuhun sivustoon sulautuva – esimerkiksi peliaulana toimiva virtuaalitila ei saisi tuntua ikävältä kompastuskiveltä, jonka kautta käyttäjien on kuljettava päästäkseen pelaamaan suosikkipelejään.

3 KONSEPTOINTI

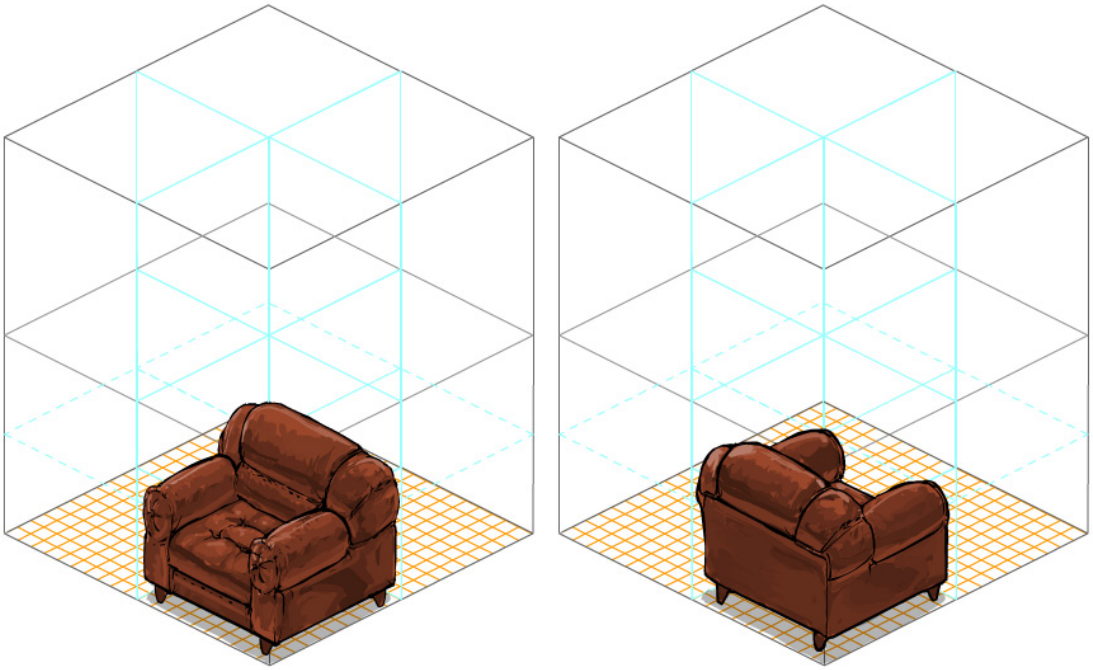
Aapelin peli- ja hahmografiikat noudattavat yhtenäistä sarjakuvamaista ilmettä, johon kuuluu selkeä musta ääriviiva, tasaiset väripinnat sekä läpinäkyvillä valkoisen ja mustan pinnoilla luodut valo- ja varjoefektit. Avatar-hahmot ovat Flashilla piirrettyjä kaksiulotteisia paperinukkeja, jotka rakentuvat erillisistä osasista; esimerkiksi jokainen kasvojen osa on erillinen elementti, joka on vaihdettavissa tai muokattavissa Aapelin hahmogeneraattorin kautta. Piirustusvaiheessa osasille on annettu yleensä yksi tai kaksi erityistä väripintaa, joiden väriä käyttäjät voivat vaihtaa mieleisekseen, joten samoistakin palasista rakennetut hahmot voivat näyttää täysin erilaisilta

Kuva 5. Aapelin hahmoja.

Kevään 2009 työharjoitteluni aikana pääsin piirtämään useita hahmovaatekokoelmia, joiden ohessa opin tehokkaasti Aapelin ilmeen vaatiman työjärjestyksen. Tutustuin samalla muihin Flash-työkaluihin ja piirustustekniikoihin, joten minulla on tämän projektin alkaessa riittävät taidot tutkia olemassa olevaa materiaalia sekä suunnitella uutta.

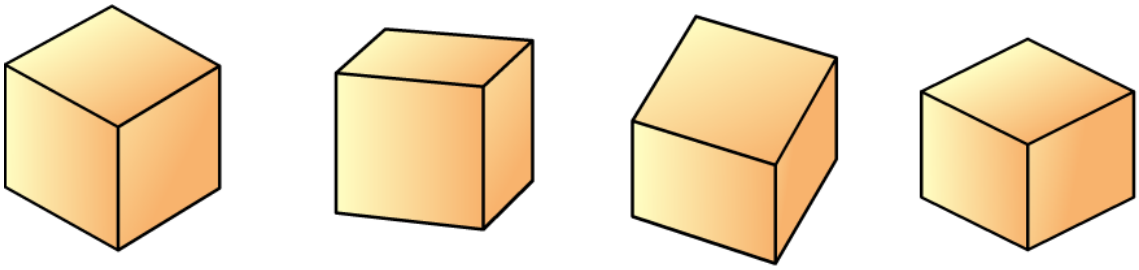
3.1 Materiaalin kartoitus

Sain Apajalta tutkittavakseni aikaisemman Playforia-projektin grafiikoiden alkuperäistiedostoja. Yli tuhannen tiedoston paketti sisälsi yli kahdenkymmenen eri teeman mukaisia settejä, esimerkiksi hotelli-, etno- ja antiikkihuonekaluja sekä seinien ja lattioiden rakennuspaloja. Jokainen esine on piirretty samalla tekniikalla kuin hahmokaupan asusteet, minkä lisäksi suuri osa huonekaluista on piirretty myös muista kuvakulmista. Koodauksen avulla on mahdollista vaihtaa huonekalujen pohjaväriä sekä muuttaa niiden suuntaa.



Kuva 6. Hotelli-huonekalusarjan nojatuoli edestä ja takaa, apulinjojen ympäröimänä.

Aksonometrinen projektio on teknisen piirtämisen muoto, jossa kolmiulotteinen kappale esitetään kaksiulotteisesti niin, että sen kukin sivu näytetään määrätyssä skaalassa. Sen variaatioita ovat isometrinen, dimetrinen ja trimetrinen perspektiivi; isometrisessä kukin kolmesta sivusta näytetään samassa skaalassa, dimetrisessä kaksi sivua, ja trimetrisessä jokaisen sivun skaala on määritelty eri tavalla (Blythe & McReynold 2005, 503).



Kuva 7. Vasemmalta oikealle: isometrinen, dimetrinen ja trimetrinen projektio. Oikeanpuolimmaisin on Playforia-projektissa käytetty projektio, joka on käytännössä litistetty versio isometrisestä.

Isometristä perspektiiviä käytetään usein teknisissä piirustuksissa, sillä sen avulla piirretyistä rakennekuvista on mm. helppo ottaa mittauksia ja arvioida osien välisiä suhteita. Vanhat 3D-pelejä edeltäneet kaksiulotteiset pelit olivat usein isometrisiä tai sitä jäljitteleviä, sillä pikseligrafiikalla on vaikea esittää puhdasta isometrisyyttä johtuen esitystavan vaatimista kolmenkymmenen asteen kulmista (mikä on nykyään mahdollista reunanpehmennyksen avulla). Kaksiulotteisilla kuvilla kyettiin jo varhain luomaan yksityiskohtaisempia ympäristöjä kuin mihin kolmiulotteinen grafiikka olisi pystynyt. Nykyäänkin monet virtuaalimaailmat ja pienpelit, esim. mobiililaitteille suunnitellut pelit, käyttävät isometristä grafiikkaa, sillä se on helppoa tuottaa ja vaatii vähän tehoja käyttäjän laitteistolta.

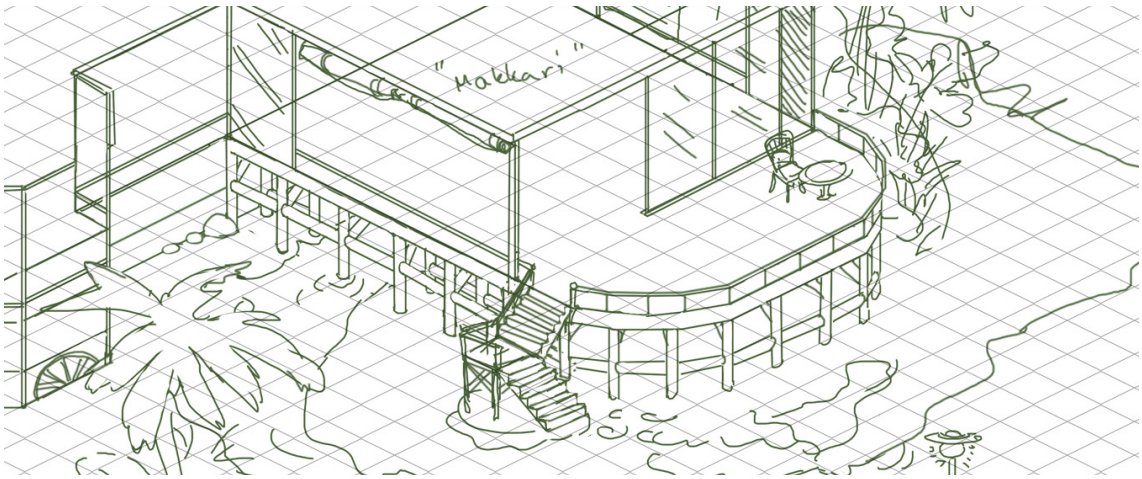
Playforia-esineet käyttävät samaa projektiota kuin Monimonsun TrunkTech-työkalu. Kummankin grafiikka perustuu ruudukkoon, joka on rakennettu kääntämällä neliötä neljäkymmentäviisi astetta ja litistämällä syntyneen kuvion korkeus puoleen. Teoriassa siis Playforia-esineet ovat siirrettävissä suoraan TrunkTech-työkaluun. Käytännössä Playforia-esineitä täytyy kuitenkin skaalata noin 75 prosenttiin alkuperäisestä, jotta ne mahtuisivat työkalun pienempään ruudukkoon, ja lisäksi niiden rakennetta on pakko optimoida rankasti, jotta ne latautuisivat nopeammin ja veisivät vähemmän käyttäjän laitteen resursseja.

3.1 Tilan konseptointi

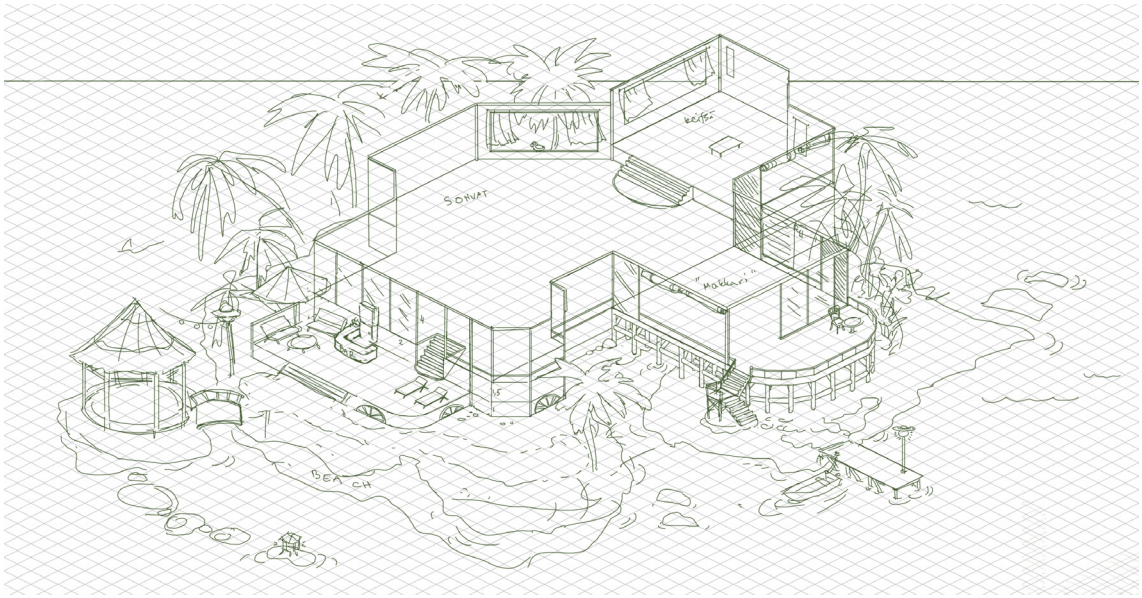
Opinnäytetyöni varsinainen graafinen työnäyte tulee olemaan testausta varten suunnittelemani tila, jonka tulisi olla omia piirustustaitojani haastava, TrunkTech-työkalua monipuolisesti koetteleva sekä ulkoisesti kiinnostava ja tyylikkään näköinen. Tutkiessani vanhaa Playforia-materiaalia sain nopeasti inspiraation trooppisesta lomasaaresta, sillä sellaisessa voisi luontevasti soveltaa lukuisia aasialais-, etno- ja viihdeteemaisia Playforia-esineitä. Lomasaari olisi hengeltään positiivinen tila, ja sen ilmeestä saisi helposti raikkaan ja värikkään, joten se olisi myös helposti markkinoitavissa. Lisäksi testisaaresta voisi myöhemmin soveltaa todellisen chat- tai pelihuoneen.

Aloitin saaren suunnittelun etsimällä kuva-aineistoa; tutkin sekä matkailutoimistojen esitteitä että Internetin kuvapankkeja. Etsin etenkin mallikuvia trooppisten saarten värimaailmasta, kasvillisuudesta sekä rakennemateriaaleista. Halusin luoda saareen erilaisia ympäristöjä, esimerkiksi hiekkarantaa, vehreää kasvillisuutta sekä lomahuvila-rakennuksia. Kirjasin jo varhaisessa vaiheessa muistiin mahdollisia Flash-ominaisuuksia, joita voisin kokeilla eri elementtien kanssa; kasvit voisin animoida huojumaan hitaasti ja vedestä voisin tehdä aaltoilevan ja kimmeltävän. Lisäksi asuinrakennuksissa voisin harjoitella teknisempää piirtämistä ja erilaisten materiaalien kuvaamista vektorigrafiikalla.

Varsinaisen luonnostelun aloitin suoraan digitaalisesti. Adobe Photoshop on minulle tuttu luonnosteluväline, joten monistin ohjelmassa valkoiselle pohjalle 80x40 pikselin kokoisen ruudukon, jolle ryhdyin vapaalla kädellä muotoilemaan lomasaaren pohjaa. Orgaaniset muodot, esimerkiksi saaren rantalinja tai kasvillisuus, eivät tuottaneet ongelmia, mutta itse rakennusten kanssa jouduin ensimmäistä kertaa ottamaan huomioon ruutukaavan. En ollut koskaan aikaisemmin piirtänyt minkäänlaista teknistä kuvaa, mutta lyhyen hapuilun jälkeen löysin sopivat työkaluasetukset. Monistamalla muutamaa perusmuotoa sain kätevästi luotua seinät ja talon perustukset.



Kuva 8. Vapaalla kädellä ruudukkopohjan päälle luonnostelemiani rakennelmia.



Kuva 9. Saari kokonaisuudessaan.

Koska koko testitila tulee koostumaan vain kaksiulotteisista kuvista sen kolmiulotteinen syvyysvaikutelma tulee luoda sijoittamalla objekteja lomittain määrätyssä järjestyksessä. Esimerkiksi talon lattia ja takaseinät kuuluvat taustakuvaan, jolloin takaseinän lähelle kävelevä hahmo näkyy aina seinän ”edessä” ja lattian ”päällä”, mutta keskelle lattiaa sijoitettu huonekalu käyttäytyy niin, että hahmo näkyy sen ”takana”, kun hahmo seisoo ruudussa, joka on huonekalun yläpuolella, mutta ”edessä”, jos se seisoo huonekalun alapuolella olevassa ruudussa. Lisäksi TrunkTech-työkalussa ns. törmäyksen tunnistus (collision detection, eli ominaisuus, joka määrittää mistä pelihahmo voi kulkea läpi ja mistä ei) määritellään merkitsemällä kokonaisia ruutuja sellaisiksi, että

hahmo ei voi kulkea niiden yli. Eli lopullisessa saaressa esimerkiksi seinien, kasvien, huonekalujen ja veden peittämät tai rajaamat ruudut pystyisin helposti merkitsemään ns. kulkukieltoon.

Kuva 10. Sekä testihahmot että seinät ja huonekalut sijaitsevat samassa tasossa, mutta työkalussa ne voidaan sijoitella niin, että syntyy illuusio siitä, että hahmo seisoo taaempana (ruudukossa ylempänä) olevien elementtien edessä, ja edessään (ruudukossa alempana) olevien asioiden takana. Punaisella rastilla merkittyihin ruutuihin hahmo ei voi kulkea, joten hahmot eivät voisi kävellä esimerkiksi pöydän tai seinien läpi.

Ruutukaavan ja törmäyksen tunnistuksen myötä jouduin ottamaan muitakin yllättäviä asioita huomioon jo luonnosvaiheessa, esimerkiksi suunnittelemani kulkureitit eivät saisi olla liian kapeita. Olin alkuvaiheessa piirtänyt vapaalla kädellä täsmälleen perusruudun levyisen pienen portaikon, mutta myöhemmässä vaiheessa huomasin, että siitä ei olisi mahdollista kulkea läpi, sillä siinä ei ollut tarpeeksi yhtenäisiä ruutuja, joista hahmo voisi kulkea. Korjasin sen kelvollisempaan muotoon, ja havainnon myötä tarkistin rakennelmieni muutkin mahdollisesti ongelmia aiheuttavat kohdat. Aksonometrisellä perspektiivillä on

helppo luoda mahdottomiakin muotoja (esimerkiksi Escherin portaikot), joten jouduin myös olemaan tarkka seinien kulmien ja korkeuserojen kanssa – eräänkin suuren piirustusvirheen huomasin vasta aivan varsinaisen työstövaiheen lopussa, mutta onneksi sen korjaaminen ei ollut vaikeaa käyttämieni symbolien ansiosta.

Kuva 11. Lopullinen Apajalle näyttämäni luonnos, johon sijoittelin alkuperäisiä Playforia-huonekaluja havainnollistamaan niiden skaalaa ja käyttömahdollisuuksia.

4 DIGITAALINEN TYÖSKENTELY

Lomasaariprojektin graafinen työskentely tapahtuu täysin digitaalisesti. Voin ohittaa kokonaan perinteisen paperille tehtävän luonnosvaiheen, sillä piirustuslaudan ja oikean ohjelmiston avulla voin piirtää lähes luonnollisesti suoraan tietokoneelle. Käytössäni ovat Adobe Photoshop, joka on käytetyin kuvanmuokkausohjelma, sekä Adobe Flash, jolla on mahdollista tehdä sekä vektorigrafiikkaa että monipuolisia animaatioita ja sovelluksia. Työvälineenä käytän Wacomin Intuos3-piirustuslautaa.

Projektin aikana työskentelen sekä vektori- että bittikarttagrafiikan parissa. Esimerkiksi lomasaaren alkuperäinen luonnos on Photoshopilla piirretty bittikarttakuva, jonka päälle rakennan Flashissa täysin vektoroidun grafiikan. Viimeistelyvaiheessa todennäköisesti osa vektorielementeistä muuntuu vielä takaisin bittikarttagrafiikaksi.

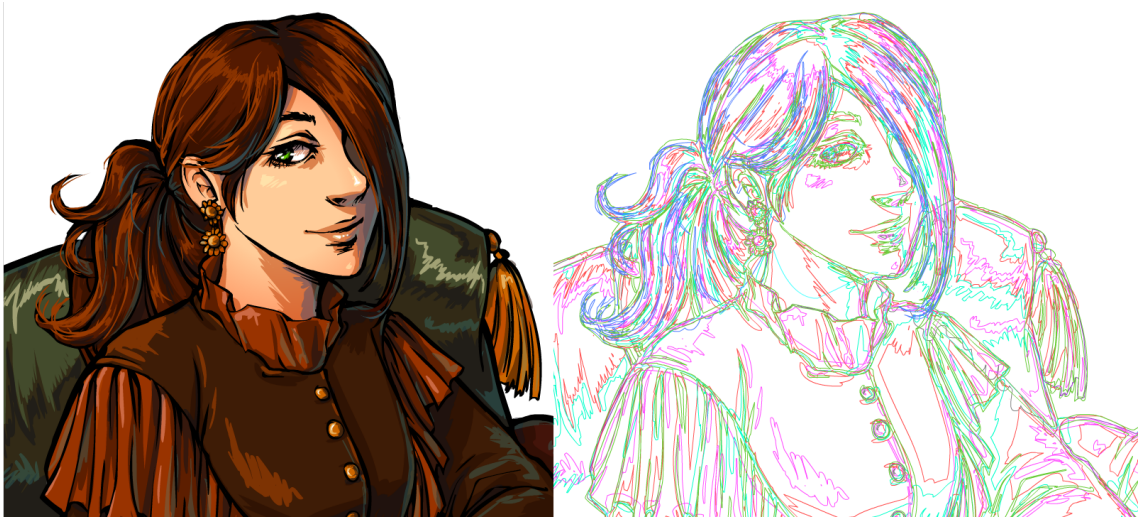
Kaksiulotteinen digitaalinen grafiikka jakautuu teknisesti bittikartta- ja vektorigrafiikkaan. Bittikarttakuva koostuu erillisistä pisteistä, eli pikseleistä. Esimerkiksi digitaaliset valokuvat muodostuvat todellisuudessa eri värisistä pikseleistä, jotka normaalikoossa katseltaessa hahmottuvat saumattomana kuvana, mutta joiden muoto paljastuu tarkasteltaessa kuvaa moninkertaisen suurennoksen avulla. Bittikarttagrafiikalla voidaan luoda monimutkaisia kuvia, mutta niiden laatu heikkenee suurennettaessa tai pienennettäessä. Vektorigrafiikka puolestaan muodostuu matemaattisten yhtälöiden avulla määritellyistä linjoista ja muodoista, minkä ansiosta se on äärettömästi skaalattavissa, mutta sen avulla on työlästä tai jopa mahdotonta saavuttaa samaa yksityiskohtaisuutta kuin esimerkiksi digitaalisissa valokuvissa.

Kuva 12. Vasemmalla on alun perin vektori-muotoisen kuvan yksityiskohdasta tehty 1000-kertainen suurennos, kun taas oikealla oleva kuva on 1000-kertainen suurennos samasta vektorikuvasta tehdystä bittikarttaversiosta. Vektoriversio säilyttää suurennettunakin tarkkuutensa, mutta bittikarttakuva ”pikselöityy” käyttökelvottomaksi.

4.1 Vektorigrafiikan piirtäminen käsin Flashissa

Digitaalinen piirtäminen on perinteisesti pyrkinyt jäljittelemään luonnollisia piirustusmenetelmiä. Nykyaikaiset piirustuslaudat voivat kertoa ohjelmalle piirtimen liikkeiden lisäksi sen kallistuskulman sekä piirtimen kärjen tunteman paineen, joiden avulla ohjelma voi vuorostaan simuloida perinteisiä työvälineitä, esimerkiksi siveltimiä, joiden viiva muuttuu piirtimen painamisen, nopeuden sekä kallistuksen myötä. Kehittyneimmätkin ohjelmat, kuten Adobe Photoshop sekä Corel Painter, tuottavat kuitenkin vain bittikarttagrafiikkaa.

Flashin yksinkertaisilla piirustustyökaluilla on mahdollista piirtää vapaalla kädellä suoraan vektorigrafiikkaa, josta on helppo tehdä niinkin yksityiskohtaista, että sitä on vaikea erottaa bittikarttakuvasta. Lisäksi valmis kuva on äärettömästi skaalattavissa, ja sen tiedostokoko on todennäköisesti pienempi kuin kuvan vastaavalla bittikarttaversiolla.

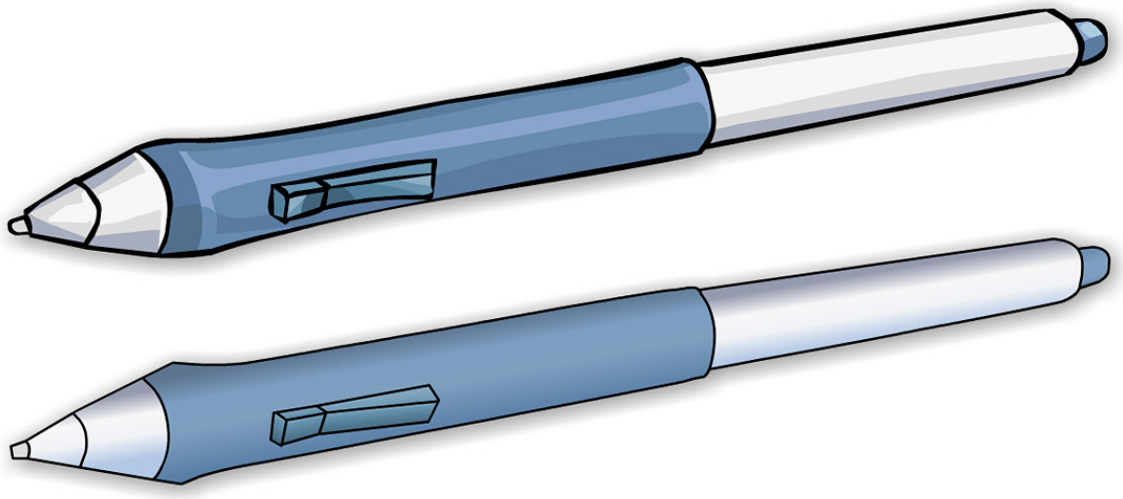


Kuva 13. Eräs kokeilukuva, jonka piirsin vuosi sitten kokeillessani ensimmäisiä kertoja Flashin piirustustyökaluja. Oikealla oleva versio näyttää kaikkien kuvan muodostavien pintojen ääriviivat.

Vektorikuvat muodostuvat ns. kulmapisteiden kautta kulkevien linjojen muodostamista viivoista ja pinnoista. Flashin piirustustyökaluista tärkein, Brush, piirtää muissa ohjelmistoissa vakiintuneista vektorityökaluista (Pen, Pencil) poiketen suoraan väripintaa, eikä viivaa. Ulkoisesti Brushin ”viivanjälki” näyttää samalta kuin esimerkiksi Photoshopin monet sivellintyökalut, sillä se tottelee piirtimen paineen tunnistusta ja sen muotoa voidaan muuttaa mm. muistuttamaan kalligrafian kapeita kärkiä. Lähempi tarkastelu paljastaa ”viivan”

kuitenkin muodostuvan lukuisten kulmapisteiden rajaamasta pinnasta. Vapaalla kädellä piirretyn pinnan muotoa voi myös muokata jälkikäteen lisäämällä tai vähentämällä kulmapisteitä sekä venyttämällä niiden välisiä kaaria. Esimerkiksi piirroshahmon ääriviivojen muokkaus muistuttaa melkein enemmän veistämistä kuin piirtämistä, sillä täydellisen tarkan ja terävän ääriviivan saavuttaminen vaatii usein lukuisten kulmapisteiden poistamista ja ääriviivojen oikaisua niin, että lopullinen muoto käyttää mahdollisimman vähäistä määrää kulmapisteitä.

Brush-työkalun jälki onkin erinomainen nimenomaan sarjakuvamaisen ilmeen tuottamiseen, ja sillä luodaankin suurin osa Aapelin kuvamaailmasta aina avatar-hahmoista peligrafiikkaan. Vektorigrafiikan luontainen selkeys ja tasaiset väripinnat sopivat hyvin pelkistetyille piirustustyyleille ja helpottavat etenkin animointia. Brush-työkalun säätöominaisuudet myös nopeuttavat piirtämisprosessia, esimerkiksi Paint Inside -määritys saa työkalun piirtämään vain tietyn pinnan sisälle, minkä avulla mm. yksinkertaisten kuvioiden varjostaminen on nopeaa eikä väriä mene ääriviivojen yli tai ulkopuolelle.



Kuva 14. Brushilla piirretyn Wacom-piirtimen ääriviiva on vaihtelevaa ja muistuttaa luonnollista tussausjälkeä. Alemman, Pen-työkalulla piirretyn piirtimen ääriviivat ovat ehdottoman tasapaksuja, mikä sopii esimerkiksi tuotekuviin tai teknisiin piirustuksiin.

Perinteisempi vektoritekniikka perustuu viivojen ja pintojen rakentamiseen kulmapiste kerrallaan. Flashin Pen-työkalu perustuu Bézier-käyrien muotoiluun kulmapisteiden välille, ja onkin tärkeä nimenomaan teknisiä piirustuksia ja kaupallista vektorigrafiikkaa tehtäessä. Viivojen leveys (ja tyyli, esim. katkoviiva) määritellään erikseen, ja käytännössä on aina tasapaksua, mikä sopiikin mm. kaavioihin ja tuotekuviin.

Väripinnat ovat tasaisia, tai niissä voi olla lineaarinen tai radiaalinen liukuväri. Yhdistelemällä tasaisia ja liukuvärisiä pintoja on kuitenkin mahdollista luoda monipuolisia värimaailmoja, ja etenkin läpinäkyvillä väripinnoilla on helppo häivyttää eri pintojen rajoja. Toisin kuin bittikarttagrafiikassa, jossa usein väripintojen reunoilla eri värit sekoittuvat toisiinsa, vektorimaailmassa väripintojen reunat ovat ehdottomia, joten niiden värien vaihtaminen lennosta sujuu helposti esimerkiksi Paint Bucket -työkalulla.

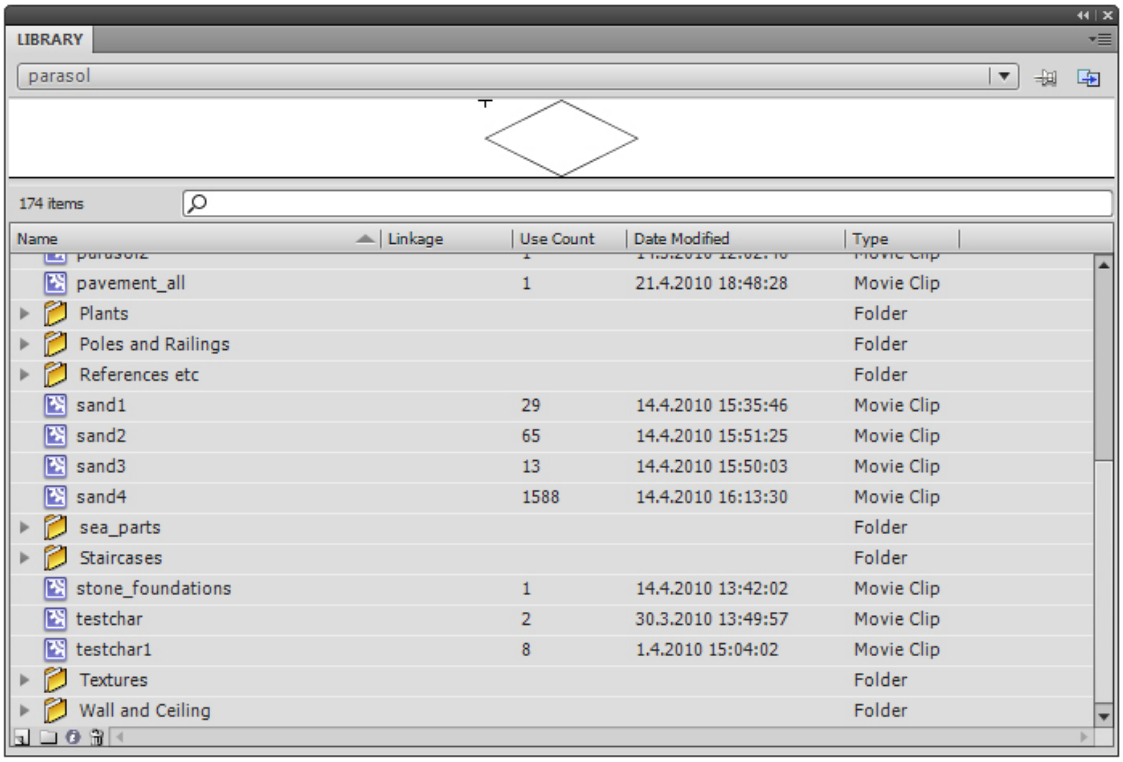
Vektorigrafiikan työstämisessä Flashilla on hyvä ottaa symbolit ja kirjasto (Library) mahdollisimman varhain käyttöön. Muuntamalla piirrettyjä elementtejä symboleiksi niiden monistaminen helpottuu; esimerkiksi puun oksan piirtäminen nopeutuu huomattavasi, jos yhden lehden piirrettyään muuntaa sen symboliksi ja sitten monistaa sitä tarvittavan määrän. Symbolien käytön selkein etu on se, että monistettunakaan symboli ei kasvata tiedoston kokoa oman yksittäisen kokonsa yli, kun taas kuvio, jossa jokainen lehti tai muu osanen olisi piirretty erikseen, kasvaisi tiedostokooltaan kunkin uuden osasen myötä.

Symbolien käyttö helpottaa myös koko Flash-esityksen hallintaa, sillä esityksessä käytössä olevia elementtejä ja tapahtumia on helpompi järjestellä symboleina – yksittäisiä polkuja tai niistä muodostettuja ryhmiä ei voi käskyttää ActionScript-koodilla. Kirjastoon tallennetut symbolit ovat turvassa vahinkomuutoksilta, mutta symbolit myös helpottavat suunniteltujen muokkausten tekoa, sillä esimerkiksi vaihtamalla kirjastoon viedyn lehtikuvion väriä kaikki esityksessä olevat kyseisen kuvion kopiot vaihtavat samanaikaisesti väriään.

Symbolimuotoja on kolme: Graphic, Button ja MovieClip. Graphic-symboleita käytetään yleisimmin yksinkertaisen grafiikan toistamiseen ja animointiin, esimerkiksi taustakuviin (Hyttinen & Lyytikäinen 2002, 86). Graphic-symbolien sisällä oleva aikajana kulkee samassa tahdissa pääesityksen aikajanan kanssa, joten niiden sisälle ei yleensä tehdä erillistä animaatiota, vaan niitä käytetään muiden animaatioiden rakennuskappaleina. Graphic-symboleita ei myöskään ole mahdollista ohjata ActionScriptillä. Button-symboleilla rakennetaan Flashin vuorovaikutteisia toimintoja, ja ne esiintyvätkin usein nimensä mukaisesti painikkeina, joilla voidaan esimerkiksi käynnistää ääntä tai animaatiota tai siirtyä toiseen osaan esitystä. Button-symbolien aikajana on erityinen, sillä se koostuu ainoastaan neljästä framesta (kuvasta), joiden kautta määritellään painikkeen eri tilojen ilme ja ominaisuudet.

Monikäyttöisin symbolimuoto onkin MovieClip, jonka sisälle on mahdollista luoda monimutkaista animaatiota (tai tuoda jopa ulkopuolinen Flash-esitys) ja joka on helposti ohjailtavissa ActionScript-koodin avulla. Esimerkiksi kaikki Aapelin hahmokaupan esineet koostuvat useista eri MovieClip-symboleista; niiden vaihdeltavat perusvärit ovat nimettyjä symboleja, joiden väriä vaihdetaan ActionSctiptin avulla, ja niiden koristeena on usein muita symboleja, esimerkiksi symboleista monistettua kangaskuviota tai kimalle-animaatioita. MovieClipien aikajana on pääesityksen aikajanasta itsenäinen, joten esim. pysäytetyssä pääesityksessä olevat MovieClip-animaatiot jatkavat pyörimistään, jos niiden sisäistä aikajanaa ei ole erikseen pysäytetty tai muutoin ohjailtu.

Symbolien lisäksi kirjaston avulla hallitaan kaikkea muutakin esityksen vaatimaa monistettavaa materiaalia – myös ääni-, video- ja bittikarttatiedostot menevät kirjastoon. Suurissa projekteissa, kuten tämän opinnäytetyön lomasaaressa, symbolien ja muun materiaalin määrä kasvaa nopeasti niin suureksi, että kirjaston selailu hankaloituu ja hidastuu. Symbolien nimeäminen ryhmittäin auttaa jo varhaisessa vaiheessa listan järjestelyä, ja samankaltaiset symbolit on myös hyvä sijoittaa omiin kansioihinsa kirjastossa.

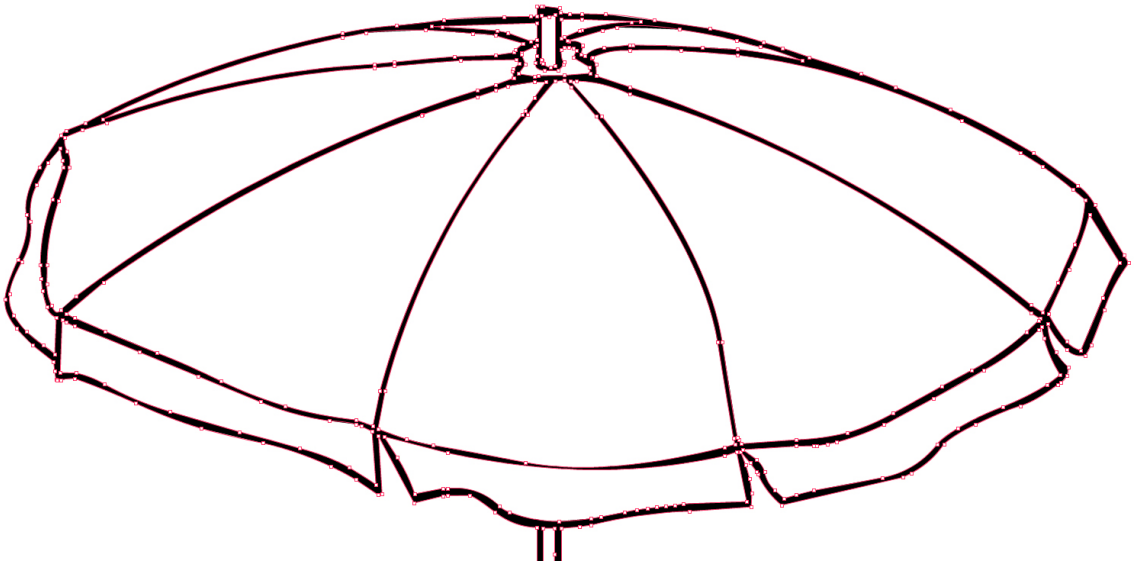


Kuva 15. Pyrin työskennellessäni pitämään kirjaston mahdollisimman järjestelmällisenä. Mielenkiintoisena yksityiskohtana kirjasto kertoo myös kunkin symbolin määrän esityksessä – esimerkiksi sand4-symbolia olen listan mukaan käyttänyt 1588 kertaa.

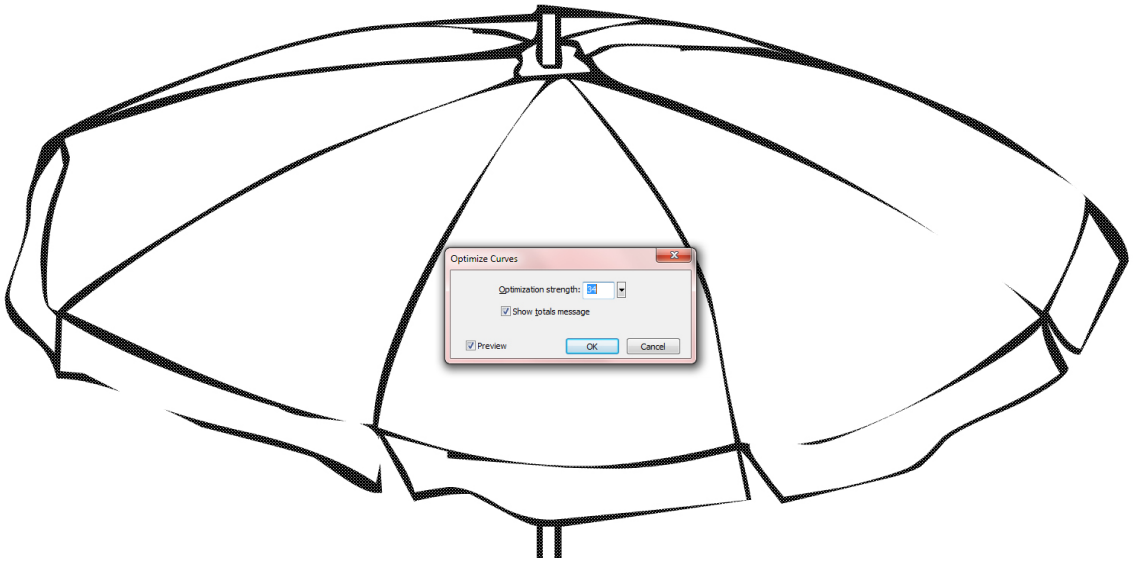
4.1 Piirtämistekniikoita

Vakiintunut digitaalipiirustusmenetelmä, joka on nykyään toteutettavissa lähes kaikissa piirtämiseen kykenevissä ohjelmissa ja jopa selaimessa toimivissa yksinkertaisissa piirustussovelluksissa, muodostuu piirtotasojen, eli layerien, avulla rakennetusta kuvasta. Piirustus alkaa usein skannatusta paperiluonnoksesta, joka tuodaan digitaalisena haluttuun ohjelmaan. Luonnoksen ylle asetetulle layerille piirretään puhtaaksi sen ääriviivat, minkä jälkeen luonnos voidaan piilottaa tai poistaa kokonaan. Ääriviiva- ja taustakuvalayerien välille luodaan useita layereita, joille piirretään kuvan värit, kuviot ja varjostus. Tekniikan avulla kuvan ääriviivat pysyvät koskemattomina omalla tasollaan, ja värialueita on helppo työstää ilman että ne sotkeutuvat toisiinsa tai menevät ääriviivojen yli.

Aapelin hahmokaupan ja vanhojen Playforia-huonekalujen piirustustekniikka soveltaa edellä kuvattua menetelmää Flashin ehdoilla. Työ alkaa vapaasti Brush-työkalulla piirretyllä luonnoksella – itse piirrän luonnoksen usein jollakin kirkkaalla värillä ja asetan sen luonnoslayerin Guideksi, eli ns. opas-moodiin, jolloin sen sisältö ei tallennu tai näy lopullisessa esityksessä. Piirrän uudelle, ylimmäksi asetetulle layerille mustalla värillä selkeät, tussimaiset ääriviivat, jonka jälkeen siistin niitä voimakkaasti sekä Smooth-, Straighten- ja Optimize Curves -toiminnoilla. Myöhempää työskentelyä helpottaa, jos ääriviivat ovat yhtenäisiä, eikä niiden välillä ole avoimia kohtia. Valkoisella valintanuolella (Subselection Tool) voin myös käsitellä yksittäisiä kulmapisteitä, ja tavoitteenani onkin niiden mahdollisimman tehokas vähentäminen. Smooth-toiminto antaa parhaat tulokset, kun sitä käyttää pieniin alueisiin kerrallaan, esimerkiksi useiden viivojen yhtymäkohdat se siistii parhaiten. Kulmapisteiden liiallinen vähentäminen, etenkin Optimize Curves -toiminnon voimakkaammilla asetuksilla, kuitenkin rikkoo helposti kuvan, joten sitä on hyvä käyttää varoen.



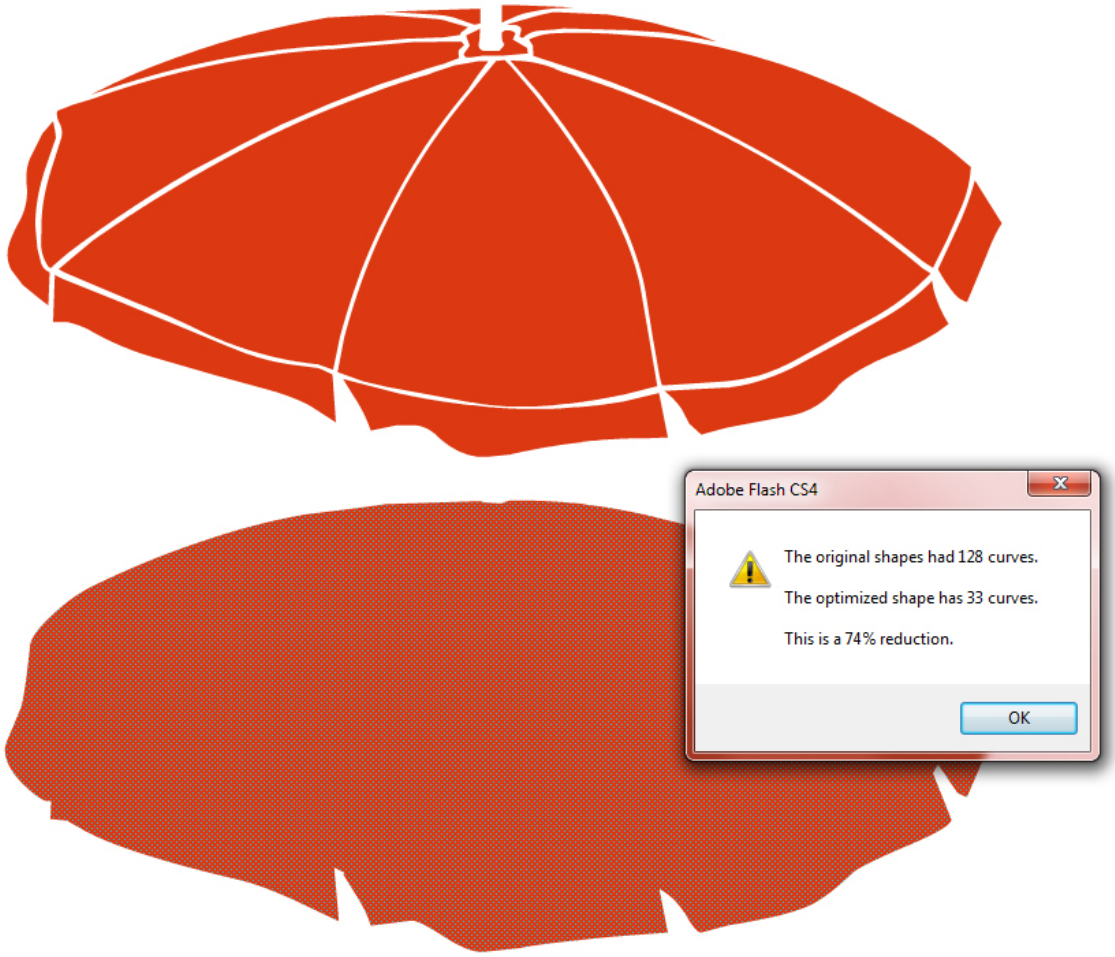
Kuva 16. Brush-työkalulla piirretyt ääriviivat eivät teknisesti ottaen ole viivoja (lines), vaan pintoja. Niitä rajaavien kulmapisteiden määrä voi helposti nousta todella korkeaksi, etenkin kohdissa, joissa useampi viiva yhtyy.



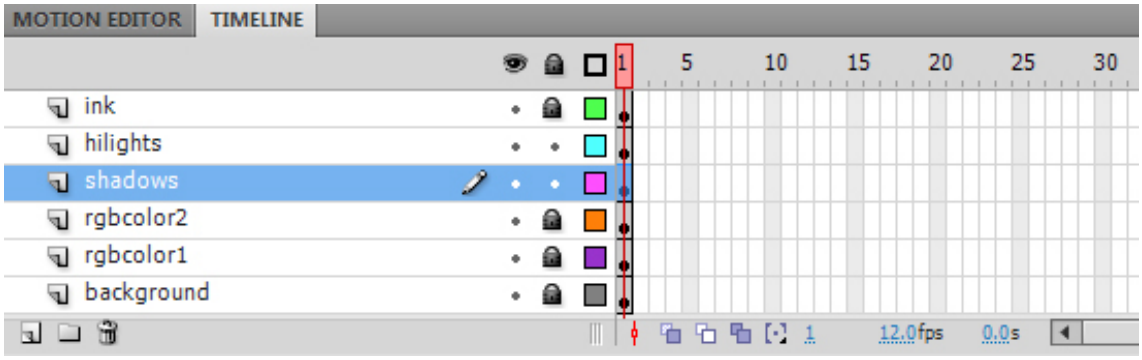
Kuva 17. Optimize Curves -toiminto vähentää tehokkaasti kulmapisteiden määrää, usein jopa monesta tuhannesta vain muutamaan sataan, mutta se myös rikkoo yksityiskohtaisia muotoja, kuten ääriviivoja. Toiminto soveltuukin paremmin laajojen pintojen, esimerkiksi tämän kuvan ääriviivojen alle tulevien väripintojen, siistimiseen ja optimointiin.

Saatuani ääriviivat valmiiksi voin nopeasti ja tarkasti värittää koko kuvan Paint Bucket -työkalulla. Saadakseni eri väripinnat erillisille layereille joudun leikkaamaan ja kopioimaan ne väri kerrallaan (Cut ja Paste in Place). Tarkoituksena on saada kuvan päävärit, joita on Aapelin esineissä yleensä kaksi, erillisille layereille, jotta ne voidaan yhdistää symboleiksi. Päävärit, jotka

muunnan symboleiksi ja nimeän koodia varten termeillä rgb1 ja rgb2, menevät omille layereilleen luonnoskuvan ja ääriviivojen väliin. Muut, vaihtumattomat värit yhdistän kolmannelle värilayerille. Jotta ääriviivojen ja väripintojen välinen sauma olisi siistimpi, kasvatan yleensä väripintojen kokoa Expand Fill -toiminnolla muutaman pikselin verran, jonka myötä väripintojen reunat sijoittuvat siististi ääriviivojen alle, ja voin vielä optimoida niitä rankasti Smooth- ja Straighten-toiminnoilla.

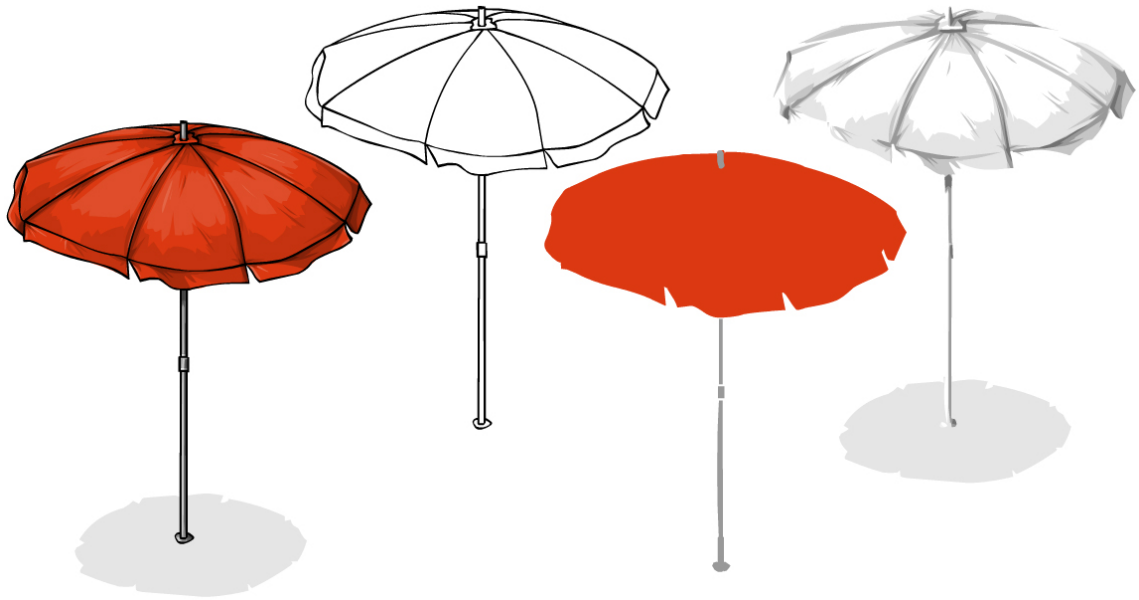


Kuva 18. Päivänvarjon ääriviivojen pohjalta tehdyt väripinnat on hyvä yhdistää yhdeksi isoksi pinnaksi, jolloin sen kulmapisteiden määrä vähenee huomattavasti. Koska pinnan reunat jäävät ääriviivojen alle piiloon, niitä on helppo optimoida voimakkaasti. Esimerkiksi Optimize Curves -toiminto tiputti kulmapisteiden määrän suoraan 128:sta 33:een kuvion muodon kärsimättä.



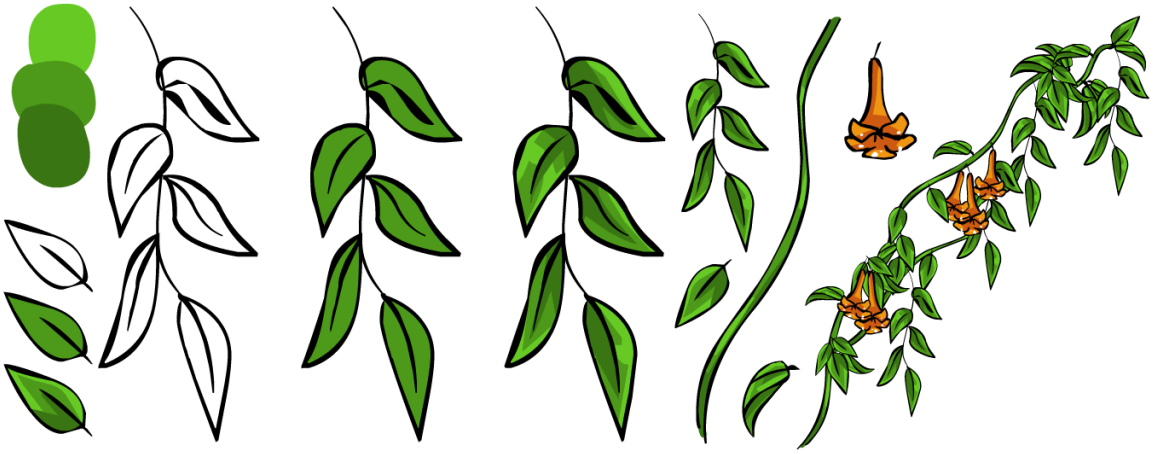
Kuva 19. Layerien järjestys yksinkertaisimmillaan. Koska kyseessä on pelkkä staattinen kuva, ei framejakaan ole enempää kuin yksi.

Väripintojen ja ääriviivojen välille luon muutaman layerin varjostusta varten. Aapelin tyyliin kuuluu selkeä mustan ja valkoisen avulla tehty varjostus, jossa pintojen läpinäkyvyyden avulla luodaan erilaisia valo- ja varjoasteita. Käytän itse yleensä 33- ja 50-prosenttista mustaa varjostuksiin, jotka piirrän vapaalla kädellä Brush-työkalulla tai leikkaan ja muotoilen Lasso-työkalulla suuremmista pinnoista. Valaistus-kohdat piirrän samalla tavalla, mutta käyttäen pohjaväreistä ja haluamastani materiaalin tunnusta riippuen erilaisia valkoisen sävyjä. Myös valo- ja varjopintoja optimoin parhaani mukaan, ja niissä voinkin käyttää jyrkempiä asetuksia kuin esimerkiksi ääriviivoja siistiessäni.



Kuva 20. Valmis päivänvarjo-grafiikka ja sen muodostavat kerrokset erikseen: ääriviivat, väripinnat ja varjostus.

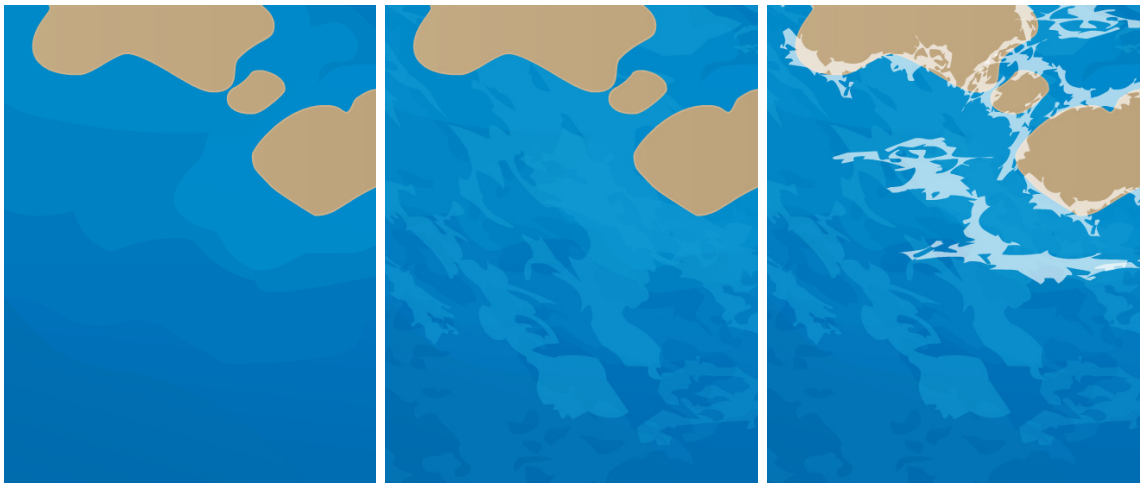
Yksinkertaisille grafiikoille, joita mm. lomasaareni on tulvillaan, käytän nopeampaa piirustusjärjestystä, joka useiden layereiden sijasta toimii yhdellä tasolla. Monet pienet ja toistuvat saaren elementit tein piirtämällä suoraan Brush-työkalulla kuvion ääriviivat, jotka siistittyäni täytin kuvion värillä käyttäen Paint Bucketia. Varjostukset piirsin suoraan väripinnan päälle Brush-työkalun Paint Inside -määrityksen avulla. Kyseinen ominaisuus muokkaa Brush-työkalua niin, että se piirtää ainoastaan sen pinnan päälle, jonka kohdalta aloitan viivan piirtämisen. Voin sen avulla piirtää huoletta yhdellä ainoalla layerilla ilman, että väriä menisi esimerkiksi ääriviivojen yli tai ulkopuolelle. Tekniikan etuna on nopeamman piirtojärjestyksen lisäksi syntyvän kuvion yksinkertaisuus, joten se voi olla helpompi ja kevyempi monistaa tai jopa animoida.



Kuva 21. Lomasaaren rönsyilevät köynnöskasvit kokosin ainoastaan kolmesta pikkuruisesta lehti-symbolista, yhdestä kukasta ja muutamasta erilaisesta koukeroisesta varresta. Useita samankaltaisia grafiikoita piirtäessäni, esimerkiksi yllä olevat köynnökset tai saaren rakennuksen monet kaiteet ja pylväät, tein itselleni yksinkertaisen väripaletin kolmesta tai neljästä väristä. Paletista on nopea kopioida värejä jokaista elementtiä varten, joten minun ei tarvinnut etsiä niitä uudestaan Flashin työläästä väriympyrästä jokaista uutta kuvaa varten.

Pienten symbolien lisäksi vektorimuotoiset tekstuurit ovat yllättävän monipuolinen väline isompien Flash-kuvien työstämisessä. Vektoritekstuurien etu on se, että niitä voi skaalata, pyörittää ja venyttää rajattomasti. Kun käytettävä tekstuuri on muutettu symboliksi (Graphic- tai MovieClip), se ei myöskään kasvata työtiedoston kokoa vaikka sitä monistaisi uudestaan ja uudestaan. Tekstuureja voi piirtää käsin tai jäljentää Flashin Trace Bitmap -toiminnolla suoraan valokuvista. Esimerkiksi lomasaaren meriveden teksturit jäljensin valokuvista, muunsin symboleiksi ja monistin monta kertaa. Tekstuuria

pyörittämällä ja venyttämällä on helppo häivyttää yksittäisten tekstuurisymbolien rajat. Myös esimerkiksi saaren nurmikko koostuu muutamasta ruohonlehtisymbolista, joista kokosin ensin perusruudukon kokoisen uuden symbolin, jolla täytin suurimmat nurmialueet, ja jonka reunamia muokkasin luonnollisemman näköisiksi lisäämällä niiden ympärille alkuperäisiä ruohonlehtiä rikkomaan ruudukon sahalaitaa.



Kuva 22. Saarta ympäröivä vesi koostuu päällekkäisistä vesitekstuureista. Pohjimmaisena on eri sävyisistä liukuväreistä muodostuva pohja, jonka päälle monistin muutamaa valokuvista jäljentämääni vesitekstuuria. Säätämällä tekstuurien läpinäkyvyyttä sekä venyttämällä ja pyörittämällä niitä sain tehokkaasti luotua tyylikkään kuviopinnan.

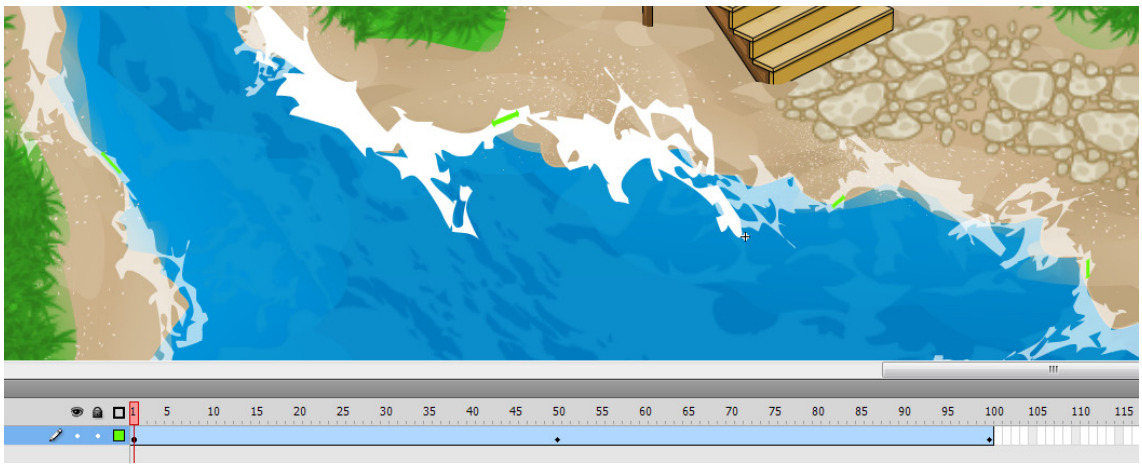
Kuva 23. Joitakin käyttämiäni vesitekstuureita, jotka kokosin useista pienistä palasista. Vaikka jäljitin alkuperäiset kuviot valokuvista, nämäkin tekstuurit ovat niin moneen kertaan muokattuja, että niillä ei enää ole mitään tekemistä lähteidensä kanssa.

4.2 Animaatio

Flashin tärkeimmät animaatiotekniikat ovat tween, Inverse Kinematics ja frame-by-frame. Tween-animaatio, joka jakautuu Flashin nykyisissä versioissa Motion, Classic ja Shape Tweeneiksi, laskee avain-framejen (keyframe) välillä tapahtuvat muutokset, ja piirtää niiden välille tapahtuvan liike-eron. Yksinkertaisin tween-animaatio syntyy kahdella keyframella, joista ensimmäiseen on sijoitettu symboli tai muu kuvio työtilan vasempaan laitaan, ja toisessa sama kuvio on työtilan oikeassa reunassa. Määrittämällä kyseisten keyframejen välille Motion tai Classic Tween saadaan aikaiseksi yksinkertainen liike vasemmalta oikealle.

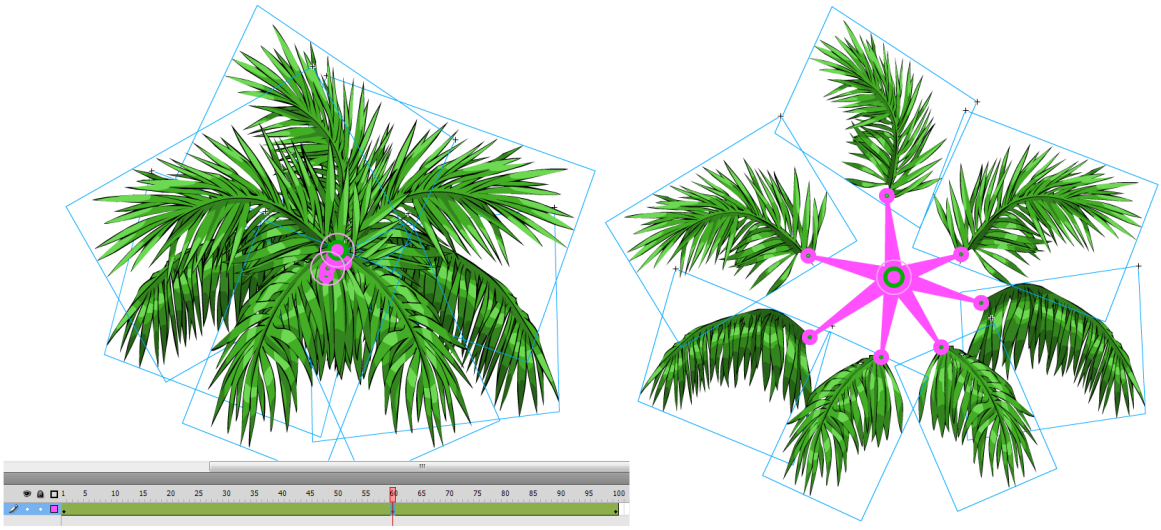
Kokeilin useita eri animaatioratkaisuja lomasaaren rantavesiä varten. Etsin inspiraatiota mm. vanhoista Muumilaakson tarinoita -piirretyistä, joissa virtaava ja aaltoileva vesi on kuvattu nerokkaasti yksinkertaisella frame-by-frame -animaatiolla. Frame-by-frame on kuitenkin työläs menetelmä, sillä nimensä mukaisesti se tarkoittaa jokaisen liikettä muodostavan kuvan piirtämistä erikseen. Frame-by-frame on erityisen epäkäytännöllinen Flashissa, sillä jokainen erillinen keyframe kasvattaa tiedoston kokoa, joten muut animaatiotekniikat ovat usein suotavampia sekä työtahdin että esityksen optimoinnin kannalta.

Lopullinen rantavesi onkin animoitu Motion Tweenin avulla. Koko saarta ympäröi sama valkoinen kuohu-symboli (tosin skaalattuna ja venytettynä), jonka sisällä on lyhyt, n. 100:n framen mittainen edestakainen liike. Kokeilin siihen alun perin myös pientä koon muutosta, mutta se ei näyttänyt hyvältä lopullisessa esityksessä. Tein Motion Tweenin avulla myös kimalle-animaation, jossa pienet valkoiset pisteet tulevat näkyville ja häipyvät pois – yhdistettynä kuohu-animaatioon ne loivat riittävän liikkeen saaren ympäröivälle vedelle. Lisäksi ainoastaan kahden symbolin käyttö koko veden animointiin säästi sekä työskentelyaikaa että piti tiedostokoon pienempänä.



Kuva 24. Kuohuanimaation Motion Tween -aikajanaa tarkastellessa itse kuvion ja sen lukuisten kopioiden yllä näkyy vihreällä niiden liikkeen suunta, minkä avulla pystyin arvioimaan koko esityksessä tapahtuvaa liikkeen kokoa ja suuntaa samanaikaisesti.

Flashin edellinen versio, Flash CS4, toi mukanaan uuden animaatiomenetelmän: käänteiskinematiikan (Inverse Kinematics, IK). Käänteiskinematiikka tarkoittaa rakennelman asentojen matemaattista laskemista lähtö- ja lopputilanteen välillä. Bones-työkalun avulla on mahdollista yhdistellä kuvia, esimerkiksi grafiikkasymboleja, ja rakentaa niistä ns. luurankoja, joissa palat ovat yhdistyneinä toisiinsa nivelten tavoin. Työkalun avulla voidaan koota erillisistä osasista esimerkiksi ihmishahmo, jolla on realistisesti kääntyvät raajat. Nivelille annettujen rajoitteiden avulla voidaan luoda uskottavia liikeratoja, jolloin hahmo voidaan esimerkiksi laittaa kävelemään luonnollisesti määrittämällä pelkästään hahmon askeleen alku- ja loppuasento.



Kuva 25. Loin palmun oksille yksinkertaisen edestakaisen liikkeen IK-aikajanalla. Oikealla ovat palmun lehdet vedettyinä erilleen, jotta kuvion ”luuranko” näkyisi paremmin. Kunkin oksasymbolin kanta on yhdistettynä keskussymboliin, joka on kuvassa vaaleanpunaisen luuranko-rakenteen alla, eikä näy myöskään lopullisessa animaatiossa.

Käytin lomasaaren palmujen animoinnissa Bones-työkalua luomaan hienoista liikettä niiden lehtiin. Yhdistin eri oksasymbolit toisiinsa keskuskappaleen kautta, jonka myöhemmin piilotin oksien alle. Näin yksinkertaisessa animaatiossa en asettanut minkäänlaisia rajoitteita oksia liikuttaville ”nivelille”. Onneksi IK-animaationi oli hyvin pienimuotoinen, sillä törmäsin pian senkin parissa Bones-työkalua vaivaaviin ohjelmointivirheisiin (bugeihin), joihin olin tutustunut aiemmin tehdessäni työharjoittelun yhteydessä monimutkaisia ihmishahmoja. Jostakin selittämättömästä syystä Bones-työkalulla tehdyt rakennelmat saattavat rikkoontua tai vääristyä peruuttamattomasti yhdestäkin valintanuolen kosketuksesta, eikä ongelmaan ole vieläkään tullut Adobelta korjausta tai edes selitystä. Ainakin Flashin CS4-version IK-animointi on vielä niin pahasti bugien vaivaama, että se on lähes käyttökelvoton muissa kuin erittäin yksinkertaisissa töissä. Kasvieni animointiin se kuitenkin soveltui hyvin, ja toivonkin Adoben korjaavan työkalun puutteet tulevissa ohjelmaversioissa.

5 OPTIMOINTI

Optimointi on parhaan vaihtoehdon etsimistä ja epäolennaisen karsimista. Tässä luvussa tarkastelen optimointia Flash-esityksen tehostamisena, jonka tavoitteena on saavuttaa mahdollisimman vähillä resursseilla toimiva, laadusta tinkimätön esitys.

Optimoinnin tavoitteena on välttää yleisimpiä ongelmia; etenkin Flash-esitykset voivat paisua tarpeettoman suuriksi tiedostokooltaan tai niiden sisältö voi käydä käyttäjän laitteistolle liian raskaaksi. Pieni tiedostokoko tietäisi lyhyempiä latausaikoja ja vähemmän rasitetta palveluntuottajan palvelimille sekä käyttäjän tietoliikenneyhteydelle (esim. mobiililaitteiden tiedonsiirto voi tulla kalliiksi ilman erityistä sopimusta). Yksinkertaisemmat esitykset toimivat useammilla alustoilla, ja niiden yhteensopivuus on parempi eri käyttöjärjestelmien ja näyttölaitteiden kanssa – mm. jollekin tietylle Flash Playerin versiolle suunnitellut Flash-esitykset voivat toimia arvaamattomasti laitteilla, joilla on asennettuna varhaisempi Flash Player -versio.

Optimointi voi parhaassa tapauksessa myös nopeuttaa työskentelyprosessia. Jos esimerkiksi Flashilla piirretyn kuvan tai animaation optimoinnin tarkoituksena on tarpeettomien elementtien vähentäminen, jotta siitä saataisiin nopeammin latautuva ja tehokkaammin pyörivä, kannattaa jo varhaisessa piirtämisvaiheessa ottaa huomioon kuvan kokoa kasvattavat ominaisuudet ja välttää niitä.

5.1 Suorituskykyyn vaikuttavat ominaisuudet

Flash-esityksiä vaivavat pullonkaulat löytyvät helposti tarkastelemalla vanhaa Flash-kirjallisuutta sekä uudemman, mobiililaitteita varten suunnitellun Flash Player Liten ohjeistusta. Vaikka tietokoneet ovat tänä päivänä moninkertaisesti tehokkaampia ja tietoliikenneyhteydet nopeampia kuin esimerkiksi 2000-luvun alussa, niiden rinnalle tulleet älypuhelimet ja muut mobiililaitteet eivät kanna sisällään samaa suorituskykyä, joten esimerkiksi 10 vuotta vanhoille tietokoneille ja modeemiyhteyksille kirjoitetut yleisohjeet ovat yhä päteviä. Flash-suunnittelija ei voi olettaa, että kaikilla käyttäjillä olisi käytössään uusinta tekniikka edustava laitteisto, ja saavuttaakseen suurimman mahdollisen yleisön on hyvä pitäytyä pienimmässä yhteisessä nimittäjässä. Kaiken lisäksi Flash Player ei aina edes osaa hyödyntää tehokkaita näytönohjaimia raskaan sisällön esittämisessä, vaikka viimeisimmässä 10-versiossa onkin tuki laitteistokiihdytykselle (Adobe 2010).

Koska Flashia käytetään pääsääntöisesti kevyen web-sisällön tuottamiseen, tiedostoissa tulisi pyrkiä mahdollisimman pieneen tiedostokokoon. Flash-esitysten, eli swf-tiedostojen, sisältö on nk. streamaavaa, eli koko tiedostoa ei ladata kerralla, vaan sen katselu voidaan aloittaa samalla kun osa esityksestä vielä latautuu. Pieni tiedostokoko takaa sen, että esityksen katselu voidaan aloittaa ilman pitkää odotusta. Kevytkin animaatio voi kuitenkin pysähtyä hetkeksi kesken soiton, jos se ei ole vielä ehtinyt ladata jotakin suurempaa osaa, esimerkiksi bittikarttaa tai ääntä, joten esityksiin on hyvä lisätä esilataaja (preloader), jonka pyöriessä esitys kerkeää lataamaan itsensä kokonaan.

Esityksen kokoa kasvattaa käytännössä kaikki sen sisältö. Jokainen yksittäinen kulmapiste, väripinta, symboli ja keyframe nostaa kilotavukokoa, puhumattakaan mukaan pakatuista bittikartta-, ääni- ja videotiedostoista. Pääsääntöisesti Flashin vektorigrafiikka on tiedostokooltaan moninkertaisesti keveämpää kuin esimerkiksi sama kuva bittikarttana, ja kuvioiden toistaminen symbolien avulla vähentää lopulliseen esitykseen tallennettavia osia. Tekstikentissä kannattaa käyttää yleisiä kirjasintyyppejä (fontteja), sillä harvinaisemmat, esimerkiksi aasialaiset fontit, on muuten pakko tallentaa mukaan swf-tiedostoon, mikä myöskin kasvattaa tiedostokokoa – koukerokirjaimien muuttaminen vektorigrafiikaksi olisi kaikista huonoin vaihtoehto, sillä lyhytkin teksti vektoripinnoiksi rikottuna sisältää valtavan määrän kulmapisteitä.

Järjestelmäfonteilla kirjoitetut tekstit sisältävätkin swf-tiedostossa vain itse tekstin muodostavat merkit sekä ohjeet, joiden avulla Flash Player piirtää tekstin oikean kokoisena ja värisenä, käyttäjän järjestelmältä löytyvää fonttia käyttäen.

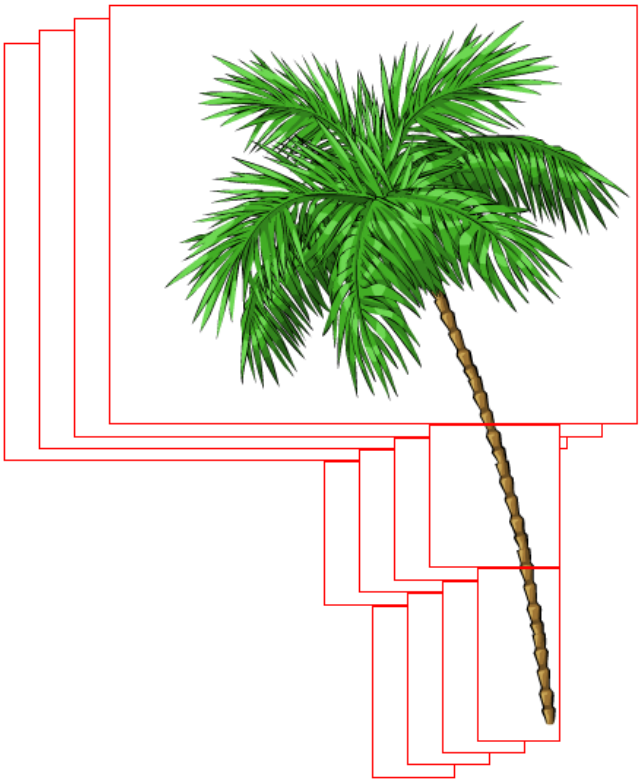
Pelkällä tiedostokoon huomioimisella ei kuitenkaan voida taata swf-tiedoston optimaalisuutta, sillä järkevän latausajan lisäksi yhtä tärkeä ehto sen toimimiselle on se, miten tehokkaasti ja vähillä resursseilla se on katseltavissa. Hyvin pienikokoisestakin Flash-esityksestä on helppo tehdä niin raskas, että siitä tulee suorastaan käyttökelvoton, sillä mitä Flash ei lataa, se piirtää lennosta, ja monimutkaisella sisällöllä on helppo ylikuormittaa vanha laitteisto.

Sham Bhangal esittää teoksessan *Flash Hacks* (2004, 322), että vaikka bittikarttojen latausaika on vektorikuvia pidempi, ne eivät käytä merkittävästi järjestelmän resursseja latauksen jälkeen, kun taas vektorikuvat ovat matemaattisen luonteensa ansiosta kompakteja, mutta laitteiston on laskettava ne uudestaan jokaisen muutoksen myötä. Esimerkiksi bittikartan liikuttelu voi joissakin tapauksissa olla tehokkaampaa kuin vastaavan vektorikuvion animointi. Käytännössä siis voi olla tilanteita, joissa tiedostokoon kasvattaminen bittikarttojen käytön takia on kannattavaa, jos niiden avulla voidaan saavuttaa parempi suorituskyky lopulliselle swf-tiedostolle kuin saman sisällön esittämisellä vektorimuodossa.

Vektorigrafiikan ongelmakohtia ovat mm. kulmapisteiden liiallinen määrä, läpinäkyvät pinnat (alpha) sekä liukuvärit (gradients). Läpinäkyvät ja liukuväriset pinnat vaativat tasaisia pintoja enemmän laskentatehoja, ja etenkin läpinäkyvä pinta toisen läpinäkyvän pinnan, liukuvärin tai bittikartan päällä voi hidastaa merkittävästi frameratea, eli sekunnissa näytettävien kuvien määrää. Varsinkin kohdassa 4.2 kuvaamani Aapelin piirustustekniikka on erittäin raskas, sillä käytännössä se sisältää usein sekä liukuvärejä että alpha-pintoja päällekkäisissä kerroksissa, mikä ei staattisessa kuvassa haittaa (esimerkiksi Aapelin hahmosivut), mutta animaatiossa jokainen uusi frame joutuisi laskemaan läpinäkyvien pintojen värit uudestaan. Jo pelkästään useiden layerien määrä jokaisessa Aapeli-hahmon muodostavassa hahmoelementissä tarkoittaa sitä, että niissä on suunnaton määrä kulmapisteitä, joista suuri osa on suorastaan turhia niiden jäädessä toisten kuvioiden alle piiloon. Flash Player nimittäin laskee jokaisessa framessa myös sellaiset elementit, jotka ovat näkymättömissä – ne voivat olla toisten elementtien

takana piilossa tai niiden peittävyys on asetettu nollaan, jolloin ne eivät erotu ollenkaan, mutta siitä huolimatta ne lasketaan mukaan frameen. Ideaalinen vektorigrafiikka koostuisikin ainoastaan tasaista väripinnoista sekä vaaka- ja pystysuorista linjoista, sillä niiden esittämiseen ei tarvita edes reunanpehmennystä (anti-aliasing), mutta monimutkaisempien efektien järkevällä käytöllä on mahdollista estää esityksen yksittäisiä elementtejä paisumasta liian raskaiksi.

Flash ei kuitenkaan aina piirrä uusiksi koko esitystä jokaista uutta framea varten. Valmiista swf-tiedostosta on mahdollista tarkastella ns. uudelleen piirrettäviä alueita (redraw regions), jotka rajaavat jokaisessa framessa tapahtuvat muutokset. Uudelleen piirrettävä alue käsittää muuntuvan objektin sekä sen peittämät muut elementit, esimerkiksi taustakuvan (Adobe 2007). Flash Playerin testi-ikkunassa on mahdollista nähdä alueet, jotka Flash piirtää uusiksi jokaista framea varten – pienempi alue vaatii vähemmän laskentatehoja. Flash Player myös yhdistää päällekkäin menevät redraw-alueet, jolloin joissakin frameissa syntyvä redraw-suorakulmio voi olla huomattavan suuri, peittäen jopa koko esityksen.



Kuva 26. Eräässä kokeessani laitoin palmun liikkumaan edestakaisin esityksen reunasta reunaan. Flash Player pyrkii sovittamaan redraw-alueet mahdollisimman lähelle esityksen muuttuvaa osaa, joka yllä olevassa esimerkissä jakaantui kolmeen suorakulmioon. Kuvassa on viiden framen aikana tapahtunut liike.

Käytännössä esityksen kutakin framea varten uudelleen piirrettävien pikselien määrä vaikuttaa suoraan siihen, kuinka nopeasti Flash Player kykenee pyörittämään esitystä (Bhangal 2004, 308). Framet, joissa koko kuva-alue päivittyy, vievät enemmän laskentatehoja kuin sellaiset esityksen vaiheet, joissa muutoksia tapahtuu vain pienellä alueella. Samoin esitykset, jotka ovat kauttaaltaan suurempia pinta-alaltaan (esimerkiksi täyden ruudun kokoon venytetty animaatio), muuttuvat myös moninkertaisesti raskaammiksi.

5.2 Vektori- vai bittikarttagrafiikka?

Kuvan muoto tulisi valita tilanteen ja Flash-esityksen tarkoituksen mukaan: vektorit soveltuvat paremmin animaatioon, jossa objektit liikkuvat, pyörivät ja niiden koko tai muoto vaihtuu. Bittikartat ovat parempia monimutkaisten kuvien esittämiseen, mutta ne kasvattavat sekä swf-tiedoston kokoa että esityksen käyttämän muistin määrää. Kysymys onkin, halutaanko pieni tiedostokoko, jolloin esitys piirretään lennosta käyttäjän koneella, vai suurempi tiedostokoko ja latausaika, jolloin esitys on suureksi osaksi rakennettu etukäteen esim. bittikarttoina.

Yleisimmät web-käytössä olevat bittikarttamuodot ovat jpg, png ja gif. Jpg soveltuu parhaiten suuria ja monivärisiä kuvia varten, ja onkin yleisin muoto valokuvien esittämiseen. Jpg:n tiedostokoko on yleensä pienin, mikä johtuu kuvan pakkauksesta – liiallinen pakkaus kuitenkin huonontaa kuvan laatua, eikä jpg kestä esimerkiksi toistuvia muokkauksia. Png on toinen yleistyvä kuvamuoto, ja sen tärkein valtti on moniasteinen läpinäkyvyys, jonka avulla png-kuvissa voi olla monimutkaisiakin häive- tai varjoefektejä. Png-kuvien tiedostokoko on usein kuitenkin huomattavasti jpg:tä suurempi, joten se soveltuu paremmin esimerkiksi pienten logojen ja muiden ikonien esittämiseen. Png on myös Adobe Fireworks -ohjelman oletusformaatti, ja sillä tehtyjä png:itä on vielä mahdollista muokata jonkin verran Flashissa. Gif-kuvien väripaletti on rajoitettu 265 väriin, joten se soveltuu lähinnä vähävärisiin kuviin, kuten logoihin. Gif-kuvilla on kuitenkin mahdollista luoda yksinkertaisia animaatioita, ja kuvamuoto on edelleen laajalti käytössä, vaikka png onkin usein laadullisesti parempi valinta.

Vektorit, jotka käytännössä vain koostuvat matemaattisista ohjeista joilla määritellään kuvioiden muoto ja väri, ovat luonnostaan pienikokoisia. Niiden selkein etu suhteessa bittikarttoihin on skaalattavuus: vektorisisältö on äärettömästi suurennettavissa ja muunneltavissa, kun taas bittikarttakuva on pikseliensä rajoittama. Esimerkiksi pienestä kuvasta suurennettu bittikartta muuttuu epäselväksi, sillä käytössä oleva ohjelma joutuu lisäämään tai korvaamaan pikseleitä arvaamalla niiden värin ja sijoittelun kuvan alkuperäisten pikselien perusteella.

Flash Playerin kahdeksas versio toi mukanaan uuden ominaisuuden; bitmap cache -toiminnon avulla Flash Player voi muuntaa MovieClip-symbolin lennossa bittikartaksi, joka säilytetään ohjelman työmuistissa yhdessä alkuperäisen vektorimuodon kanssa. Flash Player käyttää esitystä renderöidessään bittikarttaversiota vektorimuodon sijasta, jolloin sen ei tarvitse laskea vektorikuvaa uudestaan jokaista framea varten, vaan se voi pelkästään kopioida bittikarttaversion muistista suoraan esitykseen (Watson 2005). Ominaisuuden avulla on mahdollista saada huomattavakin parannus esityksen suorituskykyyn, etenkin jos elementti on staattinen tai liikkuu vain x- ja y-akseleilla (vaaka- ja pystysuunnassa, esimerkiksi taustakuva tai vieritettävä tekstikenttä), jolloin Flash Playerin tarvitsee vain liikuttaa pikseleitä ja laskea niiden alta paljastuva pinta, mutta ei luoda niitä uudestaan. Yllättäen huomasin, että itse Flash CS4:kin toimi nopeammin sen jälkeen, kun olin valtavassa työtiedostossani laittanut bitmap cachen päälle lomasaaren monimutkaisimpiin symboleihin – lopulta jopa yhdistin kaikki taustan muodostavat elementit yhdeksi symboliksi ja laitoin sille bitmap cachen päälle pelkästään nopeuttaakseni työtiedoston käsittelyä.

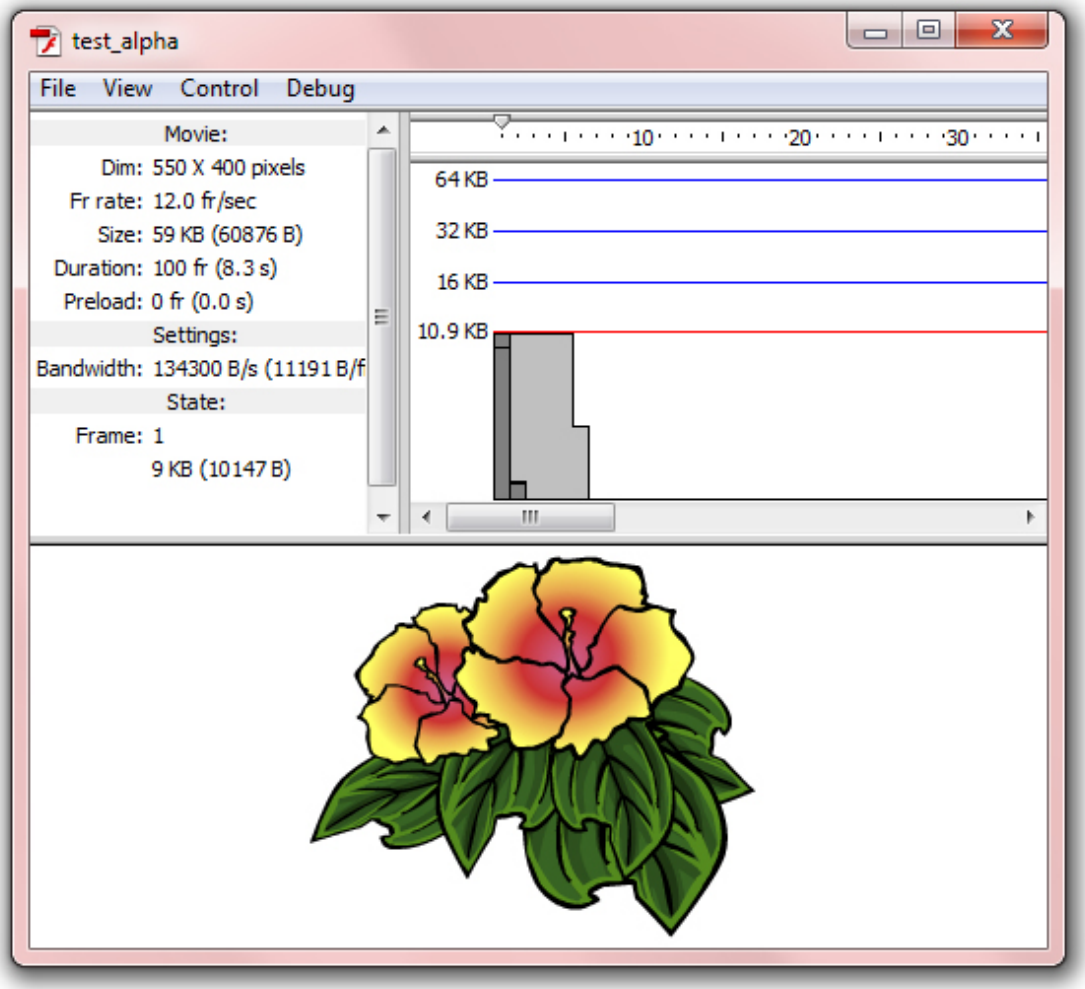
Bitmap cache lisää esityksen käyttämää muistin määrää, joten hallitsemattomasti käytettynä se voi mahdollisesti myös hidastaa käyttäjän tietokoneen toimintaa. Ominaisuus toimiikin parhaiten harkitusti käytettynä, sillä Flash Player ei käytä bittikarttaversiota luodessaan esimerkiksi jpg- tai png-kuvien pakkausmenetelmiä, joten työmuistissa säilytettävät bittikarttaversiot voivat olla suhteettoman suuria. Myös filter-efektit asettavat objekteille automaattisesti bitmap cache -tilan päälle, joten animoiduille objekteille, joilla on esimerkiksi filtereillä luotu heittovarjo, Flash Player joutuu laskemaan uuden bittikarttaversion koko symbolille jokaista uutta framea varten.

Flash-esityksen materiaalia koottaessa on hyvä arvioida kunkin objektin kohdalla, toimisiko se paremmin vektori- vai bittikarttagrafiikkana, ja onko kannattavampaa rakentaa bittikartta lennossa bitmap cachen avulla, vai luoda se etukäteen esimerkiksi jpg- tai png-kuvana, jolloin sen kokoon ja muotoon voidaan vaikuttaa pakkausmenetelmien kautta. Jpg-kuvissa ei ole mahdollista käyttää läpinäkyvyyttä, joten jpg:t ovat automaattisesti suorakulmaisen muotoisia pintoja, minkä vuoksi ne soveltuvat Flash-esityksissä parhaiten taustakuviksi. Png-kuvissa taas on mahdollista häivyttää kuvan pikselirajat moniasteisen läpinäkyvyyden avulla, joten png sopii kaikenlaisten kompaktien kuvakkeiden esittämiseen. Kun vektoriobjekti muuntuu bittikartaksi, se sovitetaan mahdollisimman tiukkaan suorakulmioon. Kompaktit muodot, esimerkiksi neliöt ja ympyrät, jättävät ympärilleen vähemmän tyhjää tilaa kuin esimerkiksi viistot kuviot. Kaikki ylimääräinen tila, vaikka se ei näkyisikään lopullisessa esityksessä, kasvattaa bittikartan kokoa. Samasta syystä kaikki bittikartat tulisi tallentaa lopullisen esityksen käyttämässä koossa, eikä niin, että esitys skaalaa niitä suuremmiksi tai pienemmiksi.

5.3 Testaaminen

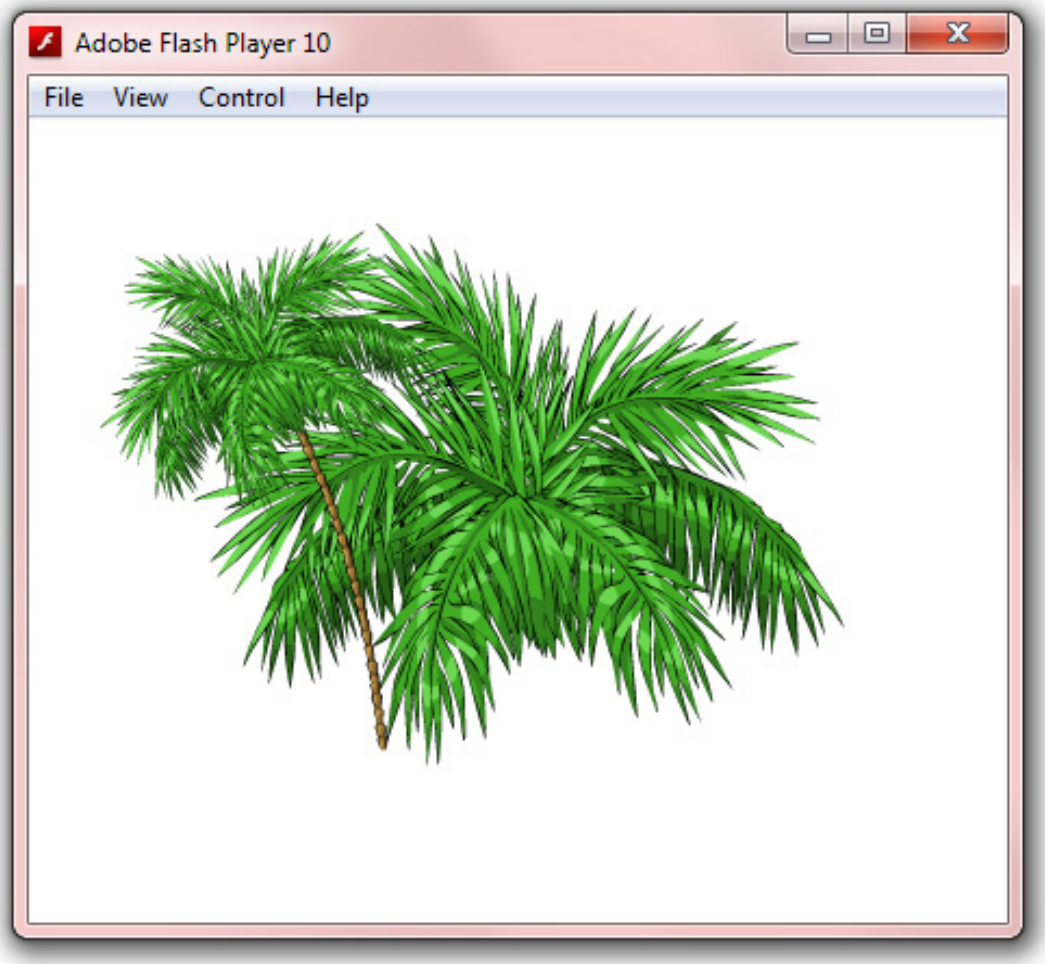
Sekä vanhat Playforia-huonekalut että Monimonsun TrunkTech-työkalu käyttävät ActionScript 3 -ohjelmointikieltä, joka vaatii vähintään Flash Player 9:n toimiakseen. Asetin kyseisen ohjelmaversion kaikkien omien tiedostojeni vähimmäisvaatimukseksi, sillä Adoben teettämän viimeisimmän käyttäjätutkimuksen mukaan noin 99 %:lla Internetin käyttäjistä on vähintään Flash Player 8 käytössään, ja noin 96 %:lla on viimeisin 10-versio (Millward Brown 2010).

Flashin testi-ikkuna näyttää kunkin framen vaatiman latausajan, ja auttaa paikallistamaan kohdat, joissa animaatio voi pysähtyä esityksen ladatessa osiaan. Lomasaari kokonaisuudessaan olisi kuitenkin ns. pysäytetty esitys, jonka kaikki osat ladataan etukäteen esilataajan aikana, joten käytin testi-ikkunan Bandwidth Profile -tilastoa lähinnä yksittäisten symbolien kilotavukoon tarkasteluun.



Kuva 27. Test Movie -toiminnolla tarkasteltu yhden symbolin sisältävä esitys. Harmaat palkit kertovat kunkin kussakin framessa ladattavan materiaalin määrän, ja niiden avulla on helppo paikallistaa animaation kohdat, joissa esitys voi hidastua, koska se ei ole vielä ehtinyt ladata kaikkea sisältöä.

Lomasaaren elementtejä suunnitellessani jouduin jatkuvasti pohtimaan, toimisiko kukin vektorimuotoinen kuvio paremmin bittikarttana vai vektorina. Testasin lomasaaren grafiikoita vertailemalla niiden tiedostokokoja sekä tarkastelemalla niiden muistin käyttöä Windowsin Tehtävänhallinnan avulla. Pyrin etenkin kokeilemaan käytännön kautta sekä Adoben ohjeistuksista että muista Internet-lähteistä löytämieni väitteiden todenperäisyyttä.



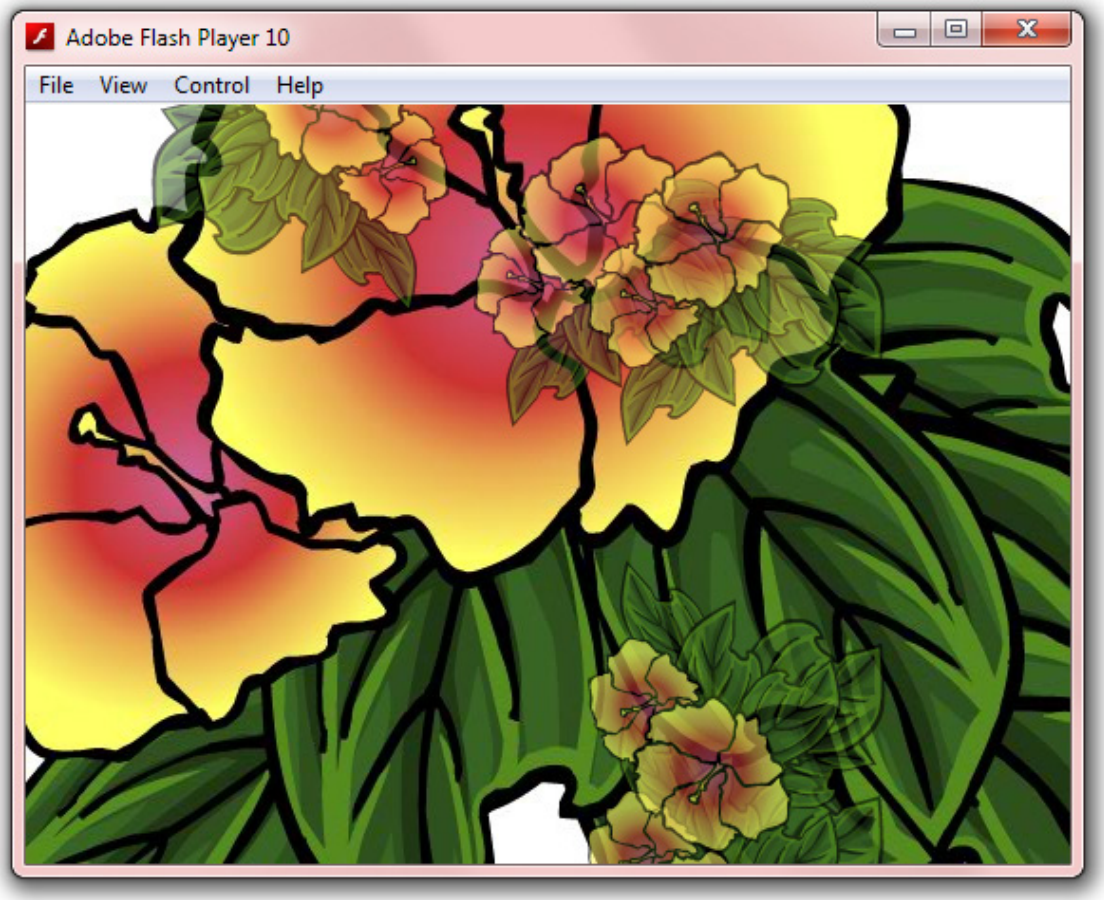
Kuva 28. Käytin testeissäni nimenomaan kaikkein monimutkaisimpia muotoja. Esimerkiksi palmujen oksat, vaikka ne ovatkin symbolien avulla kopioituja, sisältävät merkittävän määrän kulmapisteitä.

Tein useita pieniä testiesityksiä, joissa laitoin tekemiäni vektorisymboleita liikkumaan tai muuntumaan eri tavoilla, ja vertailin animaation sekä erilaisten efektien, kuten läpinäkyvyyden tai bitmap chachen, vaikutusta esityksen käyttämiin resursseihin. Ensimmäisissä testeissäni laitoin palmusymbolin liikkumaan edestakaisin esityksen reunasta toiseen niin, että se kulki samalla toisen monimutkaisen symbolin yli, ja tutkin esityksen muistin käyttöä bitmap cachen kanssa ja ilman. Koska animoituva palmu ei missään vaiheessa vaihtanut kokoa tai muotoa, pystyin laittamaan sekä siihen että taustalla olevaan symboliin bitmap cachen päälle. Tein esityksestä kaksi eri versiota, ja vertailin niitä Windowsin Tehtävänhallinnassa.

FlashPlayer.exe *32	Jenny	00	13 672 K	Adobe Flash Player 10.0 r2
FlashPlayer.exe *32	Jenny	19	13 756 K	Adobe Flash Player 10.0 r2

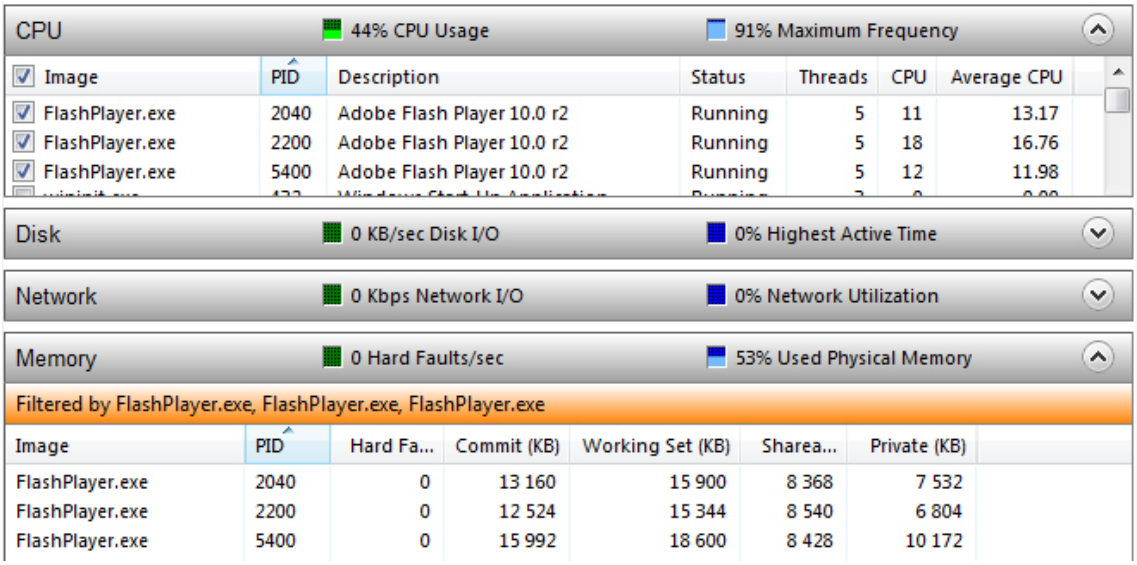
Kuva 29. Tehtävänhallinnan Prosessit-välilehti näyttää käynnissä olevien sovellusten käyttämän prosessorin ja muistin määrän (keskellä ja siitä oikealla olevat luvut).

Tehtävänhallinta osoitti selvästi, että kahdesta esityksestä ylempi, jossa oli bitmap cache päällä sekä liikkuvassa elementissä että taustassa, ei vienyt käytännössä ollenkaan prosessoritehoja ja kulutti muistiakin aavistuksen verran vähemmän. Alemmassa esityksessä, jossa en käyttänyt ollenkaan bitmap cachea, Flash Player joutui laskemaan jokaista framea varten uudestaan sekä taustan että animaation vektorimuodot, joten se myös kulutti enemmän tehoja.



Kuva 30. Monimutkaistin koetta käyttämällä elementeissä liukuvärejä ja läpinäkyvyyttä.

Toisessa kokeessa laitoin läpinäkyvät kukka-symbolit poukkoilemaan villisti ympäri esitystä monimutkaisen taustan päällä. Testasin taustakuvan muodon vaikutusta koko esityksen prosessorin ja muistin käyttöön niin, että yhdessä versiossa taustakuvana oli alkuperäinen vektorisymboli, toisessa sama symboli bitmap cache-muodossa ja kolmannessa korvasin kokonaan taustasymbolin siitä otetulla jpg-kuvalla.



Kuva 31. Windows 7:n Resource Monitor -työkalun avulla pystyin tarkemmin tarkastelemaan kunkin kokeen viemiä resursseja.

Resource Monitorin listan ylin on jpg-taustainen esitys, keskimmäinen on alkuperäinen pelkistä vektoreista koostuva versio, ja alimmassa käytin bitmap cachea taustan keventämiseen. Keskimmäinen, eli alkuperäinen vektoriesitys, vie selkeästi eniten prosessoritehoja (CPU), mutta se myös käyttää muistia vähiten. Bitmap cache -versio vie kaikkein eniten muistia, mutta vuorostaan kuormittaa prosessoria vähiten. Jpg-taustainen versio, eli listan ylin, pyörii keveämmin kuin vektoriesitys, mutta raskaammin kuin bitmap cachea hyödyntävä versio. Se kuitenkin käyttää muistia vain hivenen enemmän kuin vektoriversio.

Toisen testini myötä päättelin, että bitmap cache todellakin on hyvä pitää päällä jos se vain on mahdollista, mutta lisäksi sen viemää muistin määrää todellakin voidaan vähentää tarvittaessa käyttämällä etukäteen tallennettuja bittikarttakuvia. Jpg-taustakuva kuitenkin kasvatti itse esityksen tiedostokokoa n. 11 kilotavusta 60 kilotavuun, joten minimaalista kokoa ja nopeaa latausaikaa vaativissa Flash-sovelluksissa se voisi olla haitaksi. Käyttötarkoituksissa, joissa koko sisältö luetaan paikallisesti, esimerkiksi CD-ROM-levyt tai asennettavat sovellukset, latausajalla kuitenkaan ole merkitystä, joten bittikarttojen käyttö on niissä suotavaakin, jos niiden avulla saadaan aikaan vähempi muistin käyttö.

6 VIIMEISTELY JA TULEVAISUUS

Työni viimeinen vaihe, ennen itse virtuaalitilan kokoamista, oli sen osasten suunnitelmallinen paloittelu ja tallentaminen. Monimonsun työkalu vaatii jpg-muotoisen taustakuvan, joka yltää virtuaalitilan reunasta reunaan, ja sen päälle kootaan itse tilan rakenteet ja muut osat yksittäisistä swf- ja png-tiedostoista. Säästääkseni aikaa, vaivaa ja tiedostokokoa minun tuli saada mahdollisimman suuri osa lomasaaren elementeistä suoraan kiinni taustakuvaan. Aloitinkin jakamalla valtavan Flash-tiedostoni layerit uuteen järjestykseen; ennen olin ryhmitellyt mm. talo-, kasvillisuus- ja saarielementit omille layereilleen, mutta nyt ne tulikin ryhmitellä ne ns. syvyyden mukaisesti, eli siten, että silmämääräisesti etualalla olevat osat olisivat ylimmillä layereillä, taka-alalla olevat alemmilla, ja alimpana olevat muodostaisivat yhdistettyinä taustakuvan.

Kaikki erikseen tallennettavat elementit, puupylväistä puihin ja seiniin, syötettäisiin varsinaiseen työkaluun yksi kerrallaan, joten tavoitteenani oli yhdistellä osasia mahdollisimman paljon selkeiksi rakennuspalikoiksi. Jaoin kaikki erikseen tallennettavat osaset kahteen pääryhmään; ensimmäiseen tulivat sellaiset elementit, jotka olisivat kaiken esityksessä olevan sisällön yläpuolella, ja toiseen tulivat puoli-kolmiulotteisesti käsiteltävät elementit. Tilan rakennuspalikoiden, samoin kuin itse vanhojen Playforia-huonekalujen, erikseen syöttäminen mahdollistaisi niiden käsittelyn puoli-kolmiulotteisesti, eli niin että tiettyjen määritteiden avulla ne olisivat hahmon taka- tai etualalla riippuen hahmon sijainnista ruudukolla (kuten havainnollistin luvussa 3.2).

Monimonsun kiireistä johtuen en lopulta päässytkään opinnäytetyön aikataulun rajoissa kokoamaan lopullista lomasaartani heidän testityökaluunsa, joten joidenkin grafiikkaelementtien jakamisessa jouduin arvailemaan niiden parasta mahdollista ryhmittely- ja tallennusjärjestystä. Etenkin monet seinien nurkat aiheuttivat suunnattomasti päänvaivaa, sillä ilman käytännön kokeilua en voinut tietää, miten niistä tallennetut seinäelementit käyttäytyisivät työkalussa.

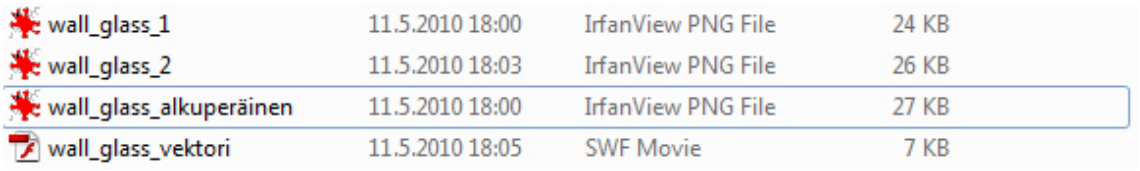
Kuva 32. Useiden elementtien lomittaisuus vaikeutti niiden ryhmittelyä.

Esimerkiksi kuvan 32 vasemmassa kuvassa testihahmot ovat Flashissa yhdellä layerillä, jonka yläpuolella olevalla layerilla on sekä vaalea että puinen seinäelementti. Jos TrunkTech-työkalu ei kykenisi erityisten määritysten avulla piirtämään seinää hahmon taakse, kun hahmo seisoo ruskealla lattialla, ja hahmon eteen kun hahmo on vaalealla lattialla, joutuisin todennäköisesti halkaisemaan seinät keskeltä kahtia ja tallentamaan ne erillisinä paloina, joilla voisin luoda tarvittavan illuusion niiden syvyyssijoittelun avulla. Oikeanpuolimmaisessa kuvassa katkoinkin portaita reunustavan kaidesymbolin kolmeen osaan samasta syystä. Alimman hahmon edessä oleva osanen sekä lasiseinään kiinnittynyt pää ovat ylemmällä layerilla, jotta näyttäisi siltä, että ne ovat alimman ja ylimmän hahmon edessä. Peittävien kaiteenpätkien välille jäävä osa on kiinni taustakuvassa, joten se jää hahmojen taakse – saumakohta näkyy keskimmäisen hahmon kohdalla.

Sain huomattavan osan saaren sisällöstä sijoitettua suoraan taustakuvaan, joten lopullisessa tallennusvaiheessa (exporttaus) minun ei tarvitsisi käsitellä erikseen läheskään jokaista rakennuspalaa. Esimerkiksi suuren osan kasveista jätin taustakuvaan, sillä ne koostuvat niin monista yksityiskohdista, että niiden tallentaminen mukaan jpg-taustakuvaan säästää paljon tilaa. Muutamat palmut otin erikseen irti, jotta voisin säilyttää niiden animaation – jokaikisen palmun ja muun kasvin animointi samanaikaisesti tulisi liian raskaaksi.

Kuva 33. Yllä saaren taustakuva, alla kaikki erikseen tallennettavat osat.

Tallensin suurimman osan erillisistä elementeistä png-muodossa, sillä esimerkiksi läpinäkyvyyttä sisältävät ikkunat ja lasiseinät sekä kiipeilevät köynnöskasvit olivat niin monimutkaisia, että ne voisivat vektorimuodossa hidastaa lopullista esitystä liikaa. Tallensin kuvatiedostoiksi tulevat elementit 24-bittisinä png:inä Flashissa, minkä lisäksi tallensin kunkin tiedoston uudestaan Save for Web -toiminnon avulla Photoshopissa, sillä sen avulla pystyin vähentämään niiden tiedostokokoa vielä muutamalla kilotavulla. Saaren lisäksi käsittelin kourallisen Playforia-huonekaluja, jotta saaren kokoamisen lopulta alkaessa minulla olisi käytettävissä kunnollinen määrä testimateriaalia.



Kuva 33. Erään lasiseinä-grafiikan tiedostokokojen vertailua eri muotojen välillä.

Kuvassa 33 alkuperäinen Flashissa tallennettu png oli kooltaan 27 kilotavua, ja sama kuva uudestaan tallennettuna Photoshopissa putosi 24 kilotavuun (kuvan 1-versio). Selittämättömästä syystä 2-versio, josta rajasin hiukan pois tyhjää tilaa kuvan ulkopuolelta, tallentui suurempana kuin 1-versio. Vertailun vuoksi sama elementti vektorimuodossa vie vain seitsemän kilotavua.

Taustakuvan tallensin yhtenä valtavana bittikarttana (bmp-muodossa) Flashissa, josta vein sen Photoshopiin ja tallensin uudestaan pakkaamattomana jpg:nä. Mittasin myös tarkasti, että taustakuva oli tarkalleen oikean kokoinen ja että se sijoittui pohjaruudukkoon oikein, sillä pikselinkin heitto väärään suuntaan aiheuttaisi sen, että tausta ei sijoittuisi oikein työkalussa.

Nimesin elementit mahdollisimman järjestelmällisesti, jotta niiden järjestely olisi myöhemmin helpompaa. Käytin samoja englanninkielisiä nimikkeitä kuin aikaisemmin symbolien nimeämisessä, esimerkiksi seinäelementit alkavat termillä wall_, kasvit plant_ ja kaikki kaiteet ja pylväät termillä pole_. Lopuksi tein vielä koko lomasaaren kokoisen ”kartan”, johon kirjasin kunkin erillisen kuva- ja vektorielementin tiedostonimen niiden oikeiden sijaintien kohdalle. Kartan avulla olisi myöhemmin mahdollista tarkistaa kunkin png- ja swf-elementin sijainti ja suhde muihin elementteihin.

Vaikka en päässytkään kokoamaan lomasaarta opinnäytetyön puitteissa, tein sen omalta osaltani kuitenkin niin valmiiksi, että kaikki elementit ovat valmiina työkaluun syöttämistä varten. Mahdollisesti ilmenevät muutokset, esimerkiksi jotkin odottamattomat työkalun vaatimat ominaisuudet tai jonkin yksittäisen elementin uudelleen tallentaminen, olisivat myös helppo toteuttaa, sillä elementtien muokkaus on vielä täysin mahdollista lomasaaren lopullisessa Flash-tiedostossa, joten niiden uudelleen exporttaaminenkin sujuu vaivatta.

7 YHTEENVETO

Kunnianhimoni vuoksi lomasaari kasvoi mitoiltaan välillä suhteettoman suureksi, ja jouduinkin jatkuvasti työn aikana karsimaan pois tai miettimään uusiksi suunnittelemiani ominaisuuksia. Kuitenkin jo varhaisessa työvaiheessa, kun olin vasta muotoillut ensimmäiset seinät ja laittanut lattiaksi tasaisen väripinnan, tunsin suurta kiintymystä luomukseeni, ja nyt valmista esitystä katsellessani voin vain nyökytellä tyytyväisenä.

Opin työn parissa korvaamattoman paljon Flashin ja sillä tehdyn grafiikan teknisestä luonteesta. Olinhan aikaisemmin osannut käyttää ohjelmaa ja sen työkaluja riittävästi yksinkertaisten kuvien ja esitysten tekemiseen, mutta tutkimukseni myötä aloin ymmärtää enemmän koko sitä prosessisarjaa, joka tapahtuu vektorikuvan piirtämisen ja sen lopullisen esittämisen välillä. Jo pelkästään piirustusvaiheessa jouduin pakottamaan itseni unohtamaan useita vakiintuneita työtapojani, jotka olivat jäänteitä digitaalimaalaus-harrastuksestani, ja opettelemaan tyhjästä vieraalta tuntuvia vektorointimenetelmiä. Ymmärtämällä paremmin eri menetelmien vaikutusta koko esityksen toimivuuteen aloin katsella muitakin Flash-esityksiä uudella tavalla.

Jouduin työn ohella ratkomaan monia mielenkiintoisia ja odottamattomia ongelmia, joita pohtiessani tulin myös huomaamattomasti avartaneeksi yleistä tietämystäni. Etenkin lähdekirjallisuutta ja -sivustoja tutkiessani törmäsin jatkuvasti uusiin kiinnostaviin Flashia sivuaviin aiheisiin, sillä opinnäytetyöhön soveltamieni neuvojen lisäksi tutustuin myös lukuisiin muihin tietolähteisiin, joilla ei ollut suoranaista yhteyttä tutkimiini asioihin, mutta joista voi olla minulle myöhemmin hyötyä muissa projekteissa. Lähdeluettelon Internet-linkkien lisäksi minulla projektin jäljiltä tallessa toinen tusina kirjanmerkkejä aiheista, jotka saattaisivat joskus olla avuksi tai joiden kautta tiedän löytäväni lisää hyödyllisiä artikkeleita.

Lomasaaren graafisia elementtejä rakennellessani jouduin myös jatkuvasti tasapainottelemaan kiinnostavien visualisten kikkojen ja web-käytön asettamien teknisten rajoitusten välillä, mikä olikin yksi päätavoitteistani projektin alkaessa. Nimenomaan toistuvan harjoittelun avulla opin arvioimaan erilaisten

ominaisuuksien painoarvoja, ja tutkimukseni myötä oppimieni neuvojen avulla onnistuinkin usein tekemään hyviä päätöksiä. Lisäksi sain hiottua aikaisempaa työskentelytapaani tehokkaammaksi, esimerkiksi olemalla järjestelmällisempi symbolien ja tiedostojen nimeämisen kanssa.

Ulkopuolinen tehtävänantaja toimi hyvänä motivoijana, vaikka sainkin käytännössä täysin vapaat kädet virtuaalitilan suunnitteluun ja toteutukseen. Olin hiukan harmistunut siitä, etten kerinnyt opinnäytetyön puitteissa kokoamaan saarta Monimonsun työkalussa ja testaamaan käytännössä työkalun ja tekemieni elementtien lopullista toimintaa, mutta toisaalta sain projektin osaltani luontevaan päätökseen, josta minulla on kuitenkin mahdollisuus jatkaa sitä myöhemmin virallisemmissa puitteissa. Olen tyytyväinen sekä valmiiseen saareeni sekä tutkimukseni myötä löytämiini uusiin oppeihin, ja olen varma, että ne ovat eduksi tulevaisuudessa sekä minulle että muille.

LÄHTEET

Adobe 2007. Adobe Flex 3 Help. Adobe Help Resource Center [verkkodokumentti]. http://livedocs.adobe.com/flex/3/html/help.html?content=performance_08.html (luettu 7.5.2010).

Adobe 2010. Display Settings. Flash Player Help [verkkodokumentti]. http://www.macromedia.com/support/documentation/en/flashplayer/help/help01.html (luettu 3.5.2010).

Bhangal, Sham 2004. Flash Hacks. E-kirja. O'Reilly Media.

Blythe, David & McReynolds, Tom 2005. Advanced graphics programming using openGL. E-kirja. Elsevier Science.

Hyttinen, Markku & Lyytikäinen, Miikka 2002. Flash MX. Jyväskylä: Docendo Finland Oy.

Millward Brown 2010. Flash Player Version Penetration. Adobe Player Census [verkkodokumentti]. http://www.adobe.com/products/player_census/flashplayer/version_penetration.html (luettu 28.4.2010).

Terdiman, Daniel 2010. Where virtual worlds once ruled, FarmVille dominates. CNET News [verkkodokumentti]. http://news.cnet.com/8301-13772_3-10460293-52.html (luettu 23.4.2010).

Tweehouse 2009. Tweehouse Blog [verkkodokumentti]. http://www.tweehouse.com/blog/ (luettu 21.3.2010).

Watson, Guy 2005. Using Bitmap Caching in Flash. Adobe Developer Connection [verkkodokumentti]. http://www.adobe.com/devnet/flash/articles/bitmap_caching.html (luettu 4.5.2010).

LIITE 1. Kuvalähteet

Kuva 1. Apaja, Playforia.
Kuva 2. Monimonsun Velocity-peli, kuvankaappaus Facebookista.
Kuva 3. Zynga, kuva http://www.farmville.com/ -etusivulta.
Kuva 4. Kuvankaappaus omalta Aapeli-sivultani.
Kuva 5. Aapeli-hahmoja Apajalta.
Kuva 6. Apaja, Playforia.

LIITE 2. Kuva koko saaresta.

LIITE 3. Yksityiskohtia saaren elementeistä.

LIITE 4. Lomarakennus sisustettuna Playforia-huonekaluilla.